UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FINAL DE CARRERA

INTEGRACIÓN DE FORMULARIOS DE GOOGLE DOCS EN LA PLATAFORMA DE GESTIÓN DEL APRENDIZAJE .LRN

AUTOR: JUAN GONZÁLEZ ADRADOS

TUTOR: ABELARDO PARDO SÁNCHEZ

octubre de 2010

Resumen

Las plataformas de Educación Virtual (*eLearning*) ofrecen herramientas tales cómo foros o videoconferencias que permiten a profesores y alumnos una mayor flexibilidad en el proceso de aprendizaje. Algunas de estas herramientas se encuentran disponibles de manera gratuita en Internet, por lo que es posible integrarlas en las Plataformas de Gestión de Aprendizaje en lugar de desarrollarlas desde cero. Una de estas herramientas es la gestión de cuestionarios, la cual permite a los profesores evaluar tanto los conocimientos de los alumnos cómo sus opiniones sobre diversos temas educativos.

Este proyecto consiste en el análisis, diseño e implementación de una aplicación que integra el servicio de formularios proporcionado por *Google Docs* dentro de la Plataforma de Gestión del Aprendizaje *.LRN*. Cómo fruto de dicha integración se abre un nuevo escenario en el que es posible añadir una funcionalidad no ofrecida por la herramienta original de formularios: la identificación correcta y segura del usuario que realiza el cuestionario.

En una primera fase, se realizó un estudio de las posibilidades de integración que ofrecen ambas plataformas. A continuación se diseñó la aplicación teniendo en cuenta las limitaciones que se habían observado en la primera fase. Tras el desarrollo de la aplicación se llevaron a cabo pruebas para garantizar su correcto funcionamiento.

Abstract

The *eLearning* platforms offer tools such as forums or videoconferencies, that allow teachers and students more flexibility in learning process. Some of these tools are available on Internet for free, so it is possible to integrate them into the Learning Management Platforms instead of developing them from scratch. One of such tools is the management of questionnaires, which allows teachers to assess both the students knowledge and their opinion on many educational topics.

This project consists of the analysis, design and implementation of software that integrates the Forms service provided by *Google Docs* within the Learning Management Platform *.LRN*. As a result of this integration, it appears a new scenario in which it is possible to add a new functionality not offered by the original tool forms: the correct and safe identification of the user performing the questionnaires.

In a first phase, it was studied the integration possibilities offered by both platforms. Then, the application was designed taking into account the limitations that had been observed in the first phase. After the development of the application, tests were conducted to ensure proper operation.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

En este capítulo se realiza una breve descripción del proyecto, así cómo del contexto en el que se ha desarrollado.

1.1. Motivación del proyecto

Con el desarrollo de las telecomunicaciones se ha popularizado en ambientes educativos el uso de Educación Virtual (*eLearning*), gracias a la cual es posible llevar a cabo el proceso educativo sin ciertas limitaciones de la educación tradicional entre las que destaca la necesidad de que profesor y alumno se encuentren en un mismo lugar al mismo tiempo.

Las Plataformas de Gestión de Aprendizaje (*LMS*) ofrecen una serie de herramientas en función de las necesidades y la capacidad tecnológica. En los últimos años, muchos de estos servicios se encuentran disponibles en Internet de manera gratuita, por lo que ya no es necesario desarrollar dichas herramientas, sino integrarlas dentro de la plataforma.

Una de estas herramientas son los cuestionarios (también llamados formularios), los cuales consisten en una serie de preguntas de distintos tipos, a través de las cuales es posible obtener y procesar información de los usuarios. Esto permite a los profesores evaluar tanto los conocimientos de los alumnos cómo sus opiniones sobre diversos temas educativos.

Google ofrece un servicio de diseño y realización de formularios online dentro de su paquete *Google Docs*, el cual en su versión gratuita[1] no ofrece la opción de identificar a los usuarios que realizan las respuestas, de manera que cualquier persona tiene la posibilidad de realizarlos de

[1] Existe la posibilidad de realizar formularios autenticados para usuarios de Google Apps (versión no gratuita).

manera anónima. Los datos recopilados por estos cuestionarios son en cambio confidenciales.

Dentro del gran abanico de Plataformas de Gestión de Aprendizaje, una de las principales es *.LRN*. Esta plataforma permite gestionar una serie de cursos pertenecientes a uno o varios centros educativos, cada uno de los cuales tiene sus propios usuarios (administradores, profesores y alumnos). Para cada curso se facilitan una serie de herramientas o aplicaciones (también llamados paquetes), las cuales pueden ofrecer sus funciones teniendo en consideración la identidad del usuario que está identificado en el sistema. La plataforma *.LRN* es de código abierto (*OpenSource*) y ofrece la posibilidad de que cualquier desarrollador pueda crear nuevos paquetes.

El paquete *Google Docs* ofrece mediante dos herramientas la posibilidad de realizar de formularios. En primer lugar, *Google Formularios* permite el diseño de los cuestionarios y posibilita su realización, pero carece de un *API* que le permita integrarse oficialmente dentro de otras plataformas. *Google Spreadsheets* por su parte se encarga de la recopilación y procesamiento de las respuestas obtenidas mediante la herramienta anterior, permitiendo el acceso e integración con otras plataformas mediante peticiones *HTTP*.

En este proyecto se ha desarrollado un paquete para *.LRN* que integra la herramienta de *Google Formularios*, añadiendo la identificación de los usuarios de *.LRN* a cada respuesta, y restringiendo su acceso a los usuarios pertenecientes a la comunidad en la que se instala el paquete que no hayan realizado un número determinado de veces el formulario, y siempre y cuando se envíe el cuestionario dentro de un plazo estipulado.

El principal problema que resuelve el paquete desarrollado es conseguir la identificación de los usuarios, con toda la seguridad que ello implica: no suplantación de identidad por parte de otra persona, ni repudio por parte del usuario que realizó el cuestionario.

1.2. Objetivos

Los objetivos principales del proyecto son: integrar el servicio de realización de cuestionarios de *Google Formularios* dentro de la plataforma *.LRN*, conseguir una identificación segura de los usuarios, poder filtrar las respuestas que no procedan de usuarios de la plataforma y permitir el manejo de la información desde dentro de la aplicación.

La realización de estos objetivos principales implica el desarrollo de los siguientes objetivos parciales.

- Estudiar en detalle las dos plataformas involucradas en el proyecto: *.LRN* y *Google Docs*.

Para ello fue necesario analizar la documentación existente, así cómo realizar pruebas con ambas plataformas, utilizando las herramientas necesarias. Para el caso de *.LRN* fue necesario estudiar la plataforma *OpenACS*, aprender a programar utilizando el lenguaje *TCL*, conocer su sistema de plantillas y el modelo de objetos en la base de datos. Además fue necesario buscar herramientas disponibles para dicha plataforma, que posibilitaran el tratamiento de documentos *XML* y la realización de peticiones *HTTPS* a través de conexiones seguras.

- Diseñar la aplicación considerando el problema a resolver y las limitaciones que ofrecen las tecnologías estudiadas. Esto ha supuesto diseñar la base de datos, determinando las tablas, sus atributos y sus relaciones; y también estructurar el código, determinando qué páginas y procedimientos desarrollar, así cómo la interacción que deben tener entre sí.

- Desarrollar el código necesario para la aplicación, de una manera estructurada y documentada. Esto se ha llevado a cabo de una manera incremental, realizando en un principio versiones sencillas que han permitido probar el correcto funcionamiento del código, y su correcta interacción con la base de datos y con *Google Docs*.

- Realizar pruebas que permitan comprobar el correcto funcionamiento de la aplicación final.

- Documentar el trabajo realizado en la presente memoria.

Estos objetivos se cubrieron a lo largo de distintas fases del desarrollo, aunque no de manera completamente secuencial, siendo necesario volver estudiar elementos que pasaron desapercibidos y rediseñar la aplicación.

1.3. Descripción

En esta sección se describe de manera básica las características de la aplicación, los usuarios y plataformas que intervienen en ella, y las limitaciones existentes.

1.3.1. Características básicas

Las dos características básicas que ofrece la aplicación son:

- Permitir realizar formularios únicamente a usuarios autenticados en la comunidad de *.LRN* en que se encuentra la aplicación, siempre que sea dentro del plazo del formulario y no se hayan alcanzado el número máximo de intentos.

- Permitir a los administradores (profesores) acceder a la lista de respuestas de los formularios, en la que se reconozca qué usuario ha realizado cada respuesta.

1.3.2. Modelo de usuario

Existen dos tipos de usuarios distintos en la aplicación: los que tienen permisos de administración y los que no.

- Usuarios sin permisos de administración. Se encargan de realizar los cuestionarios a través de la aplicación. En el entorno de *eLearning* se corresponderían con los alumnos.

- Administradores. Se encargan de instalar la aplicación en una comunidad de usuarios, añadir y modificar los parámetros de los formularios de dicha comunidad, y acceder a las respuestas de los formularios. Estas funciones pueden ser llevadas a cabo por una o varias personas, que dentro de la plataforma *.LRN* han de tener permiso de administración. La función de poder acceder a las respuestas autenticadas resulta de especial importancia para un profesor, el cual puede evaluar a los alumnos en función de las respuestas o acceder a las estadísticas de estas.

1.3.3. Arquitectura

El paquete se ha desarrollado sobre la plataforma *.LRN*, utilizando además la plataforma *Google Docs*. Al utilizar la aplicación, el usuario de tipo administrador accede a ambas plataformas. Aquellos usuarios que no sean administradores acceden únicamente a *.LRN*, dentro de la cual, la aplicación desarrollada accede a *Google Docs*.

La plataforma *Google Docs* se encarga de almacenar tanto la estructura de los cuestionarios, es decir, las preguntas (texto y tipo) que las componen; cómo las respuestas que realizan los usuarios a estos cuestionarios.

Por su parte, la plataforma *.LRN* se encarga de mantener un registro de los usuarios y permitirles el acceso a los formularios. Además, lleva un registro tanto de los cuestionarios cómo de las respuestas, en el que almacena únicamente parámetros adicionales de cada formulario que

restringen su acceso (comunidad de usuarios, plazo temporal y veces que se puede realizar) para el caso de los formularios, y la identificación de cada usuario para el caso de sus respuestas.

En la figura 1.1 se muestran las dos plataformas utilizadas por la aplicación, así cómo los dos tipos de usuarios existentes, y las interacciones entre todos estos elementos.

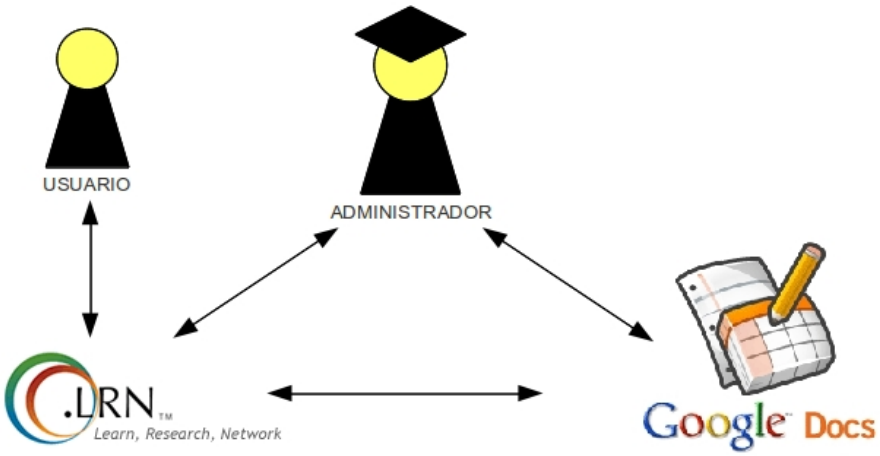


Figura 1.1: Arquitectura de la aplicación.

1.3.4. Limitaciones

En el desarrollo del proyecto ha resultado determinante el hecho de que el servicio *Google Formularios* no dispone de un *API* para acceder y modificar las características de los formularios. Por ello, el acceso a *Google Docs* desde el paquete de *.LRN* se ha llevado a cabo utilizando el *API* del servicio *Google Spreadsheet* (el cual se integra con *Google Formularios* para almacenar las respuestas), y realizando peticiones *HTTP* que simulen el acceso que realizaría un usuario a *Google Formularios*, a través de su navegador utilizando la interfaz proporcionada por *Google*.

Este hecho ha supuesto la limitación de que la aplicación no pueda acceder a la estructura de un formulario y modificarla para que este disponga de un campo adicional, en el cual se lleve a cabo la identificación del usuario. Por tanto, el usuario administrador debe encargarse de reservar un campo para llevar a cabo dicha identificación, e introducir a la aplicación el nombre de dicho campo.

Esto también implica un riesgo importante para la aplicación, ya que *Google* podría introducir cambios en el funcionamiento de los formularios que supusieran que la aplicación dejara de

funcionar, a menos que esta se actualizara para adaptarse a dichas modificaciones.

Otra limitación existente en la aplicación se debe al hecho de que los formularios de *Google* ofrecen la posibilidad de estar formados por más de una página, y mostrar unas páginas u otras en función de las opciones seleccionadas por el usuario. La aplicación resulta incompatible con este tipo de Formularios y su complejidad requeriría un diseño completamente distinto al implementado. Se ha tenido en cuenta que ante la posibilidad de que el usuario añada un formulario de este tipo, la aplicación no experimente ningún funcionamiento no deseado. Ante tales formularios la aplicación funcionaría guardando únicamente las respuestas de la última página.

1.4. Contenido del documento

Este documento ha sido organizado en cinco capítulos, precedidos de un resumen (*abstract*), y seguidos por cuatro apéndices. El primer capítulo, es decir, el presente, ha ofrecido una breve introducción del proyecto.

El siguiente capítulo se ocupa del estado del arte, describiendo tanto los Entornos Virtuales de Aprendizaje cómo las herramientas de cuestionarios *online*. Las tecnologías implicadas en el proyecto se detallan en el tercer capítulo, argumentando las razones que han llevado a elegir el uso de algunas herramientas en detrimento de otras.

El principal capítulo es el cuarto. En él se describe la aplicación, cuál es su estructura y cómo se ha diseñado teniendo en cuenta las posibilidades y limitaciones de las tecnologías utilizadas que fueron descritas en el capítulo anterior.

En el último y quinto capítulo se exponen tanto las conclusiones a las que se ha llegado después de realizar el trabajo cómo las posibles ampliaciones del proyecto que podrían llevarse a cabo.

Por último, en los apéndices se detalla el presupuesto del proyecto, el juego de pruebas utilizado para verificar el correcto funcionamiento de la aplicación, así cómo los manuales de instalación y de usuario.

Capítulo 2

Estado del arte

En este capítulo se describe la situación actual de los Entornos Virtuales de Aprendizaje, la importancia del *eLearning* en la sociedad actual, las diferencias entre las distintas opciones actualmente disponibles en el mercado, así cómo las tendencias en el desarrollo de dichos entornos.

También se describirán qué son las plataformas *online* de cuestionarios, qué funcionalidades ofrecen y las diferencias existentes entre las distintas alternativas disponibles.

En este capítulo no se tratarán en detalle cada una de las tecnologías subyacentes existentes en cada una de estos dos tipos de plataformas. La descripción y análisis de todas las herramientas y lenguajes de programación utilizados en el proyecto se llevará acabo en el capítulo posterior.

2.1. Entornos Virtuales de Aprendizaje (LMS)

La Educación Virtual o *eLearning* consiste en la formación a distancia utilizando para ello fundamentalmente Internet. La principal aportación de este tipo de educación frente a la tradicional es la flexibilidad, tanto espacial cómo temporal. Esto se traduce en una serie de ventajas, entre las que podemos enumerar:

- Fluidez en la comunicación. La interacción con profesores y otros compañeros se puede llevar a cabo mediante herramientas síncronas (tales cómo *chat* o videoconferencia) o asíncronas (*email* o foro). Estas herramientas permiten que exista una mayor interactividad al suprimirse las barreras geográficas y temporales (en este caso si las herramientas son asíncronas).
- Personalización, ya que cada estudiante puede llevar distinto ritmo de aprendizaje.

- Monitorización. Es posible realizar un seguimiento del proceso de aprendizaje del alumno.
- Mayor riqueza de contenido. En la Educación Virtual se pueden emplear distintos formatos tales cómo sonidos, vídeos, animaciones y demás aplicaciones interactivas que estarían muy limitadas en la educación tradicional.
- Mayor accesibilidad. La Educación Virtual es capaz de eliminar barreras para alumnos con discapacidades visuales, auditivas o motoras.
- Menores costes. Los costes del despliegue y mantenimiento de la infraestructura que posibilita el *eLearning* resultan en la mayoría de las ocasiones inferiores al coste de la enseñanza en las aulas.

La utilización del *eLearning* también tiene algunos inconvenientes que podríamos señalar:

- Necesidad de aprendizaje de nuevas tecnologías. Tanto alumnos cómo profesores deben adquirir destreza con las herramientas utilizadas.
- Dependencia de medios tecnológicos. El equipo informático se convierte en algo imprescindible. Además, la mayoría de las herramientas requieren una permanente conexión a internet, y algunas de ellas con unos requerimientos considerables (por ejemplo la Videoconferencia requiere un mínimo de ancho de banda).
- Resistencia cultural. Algunos sectores se muestran reacios a la utilización de esta tecnología. En algunas ocasiones a causa de simples prejuicios, pero en muchas otras debido a un pobre nivel de conocimientos.

Un Entorno Virtual de Aprendizaje (*Learning Management System*, en adelante *LMS*) es una plataforma de *eLearning* que permite la administración, documentación y monitorización de cursos, grupos, eventos y cualquier contenido de aprendizaje.

La mayoría de los *LMSs* tienen Sistemas de Gestión de Contenidos (*Content Management System*, en adelante *CMS*) que en lugar de estar orientados exclusivamente a generar contenidos genéricos, están especializados en contenidos de *eLearning*. A dichos *CMSs* especializados en *eLearning* se les conoce cómo *Learning Content Management System* (*LCMS*). Conviene aclarar que aunque en algunas publicaciones son considerados equivalentes los términos *CMS*, *LCMS*

y *LMS*, un *LCMS* puede constituir la parte fundamental de un *LMS*, pero estrictamente no es todo el *LMS* sino la parte de gestión de contenidos.

Un *LMS* debe contener fundamentalmente las siguientes herramientas [15]:

- Herramientas de administración. El administrador del sistema tiene que ser capaz de gestionar usuarios, con sus roles correspondientes; cursos, asignándolos a los correspondientes usuarios, y un sistema con la configuración necesaria.
- Herramientas de comunicación, tanto asíncronas (*emails*, foros, etc.) cómo síncronas (*chats*, videoconferencia, etc.).
- Herramientas de evaluación y valoración.
- Herramientas de creación, integración y acceso a contenidos.
- Herramientas de seguimiento del aprendizaje de los alumnos.

2.1.1. Plataformas existentes

Existen una gran cantidad de *LMS* disponibles, la gran mayoría *Open Source*, aunque también tienen una importante presencia en el mercado las aplicaciones comerciales. Entre los *LMS* más utilizados podemos señalar:

Blackboard

Blackboard Learning System[1] ha sido creado por *Blackboard Inc.*, una empresa que surgió en Washington DC en 1998 de la unión de *Blackboard LLC* y *CourseInfo LLC*. En 2009 la empresa adquirió *Angel Learning Inc.*, la cual era responsable del *LMS ANGEL*. La última versión de *Blackboard* (9.1) fue lanzada al mercado a finales de 2009.

Este *LMS* es un producto comercial, de pago, y cerrado, cuyo coste medio es de alrededor de 50.000 dólares anuales [18]. El hospedaje y mantenimiento del portal puede ser llevado a cabo por el cliente, o puede ser contratarlo a *Blackboard Inc.*.

Existen aproximadamente 3000 sitios que utilizan *Blackboard* en cualquiera de sus versiones, si bien, la tendencia del número de usuarios es a la baja, observándose una disminución en

[1] http://www.blackboard.com/

los últimos años [17]. La fiabilidad de *Blackboard* ha sido criticada por un gran número de instituciones y usuarios, que han experimentado numerosos *bugs* [21] [9].

Blackboard Inc. ha registrado numerosas patentes de *software* en Estados Unidos, incluyendo el propio concepto de *LMS* [16], lo cual fue posteriormente invalidado por la Oficina de Patentes Estadounidense [8]. Así mismo, ha realizado demandas por violaciones de dichas patentes contra *Desire2Learn*, *aTutor* y *Sakai*, que se resolvieron en su contra, al considerarse inválidas [7].

Desire2Learn

Desire2Learn ha sido desarrollado por *Desire2Learn Inc.*[2] una corporación formada en 1999 en Canadá. Se trata de un producto comercial, cerrado y de pago, cuyo coste es inferior a *Blackboard* [6]. Tiene una menor cuota de mercado que *Blackboard* [10], aunque se está incrementando [11].

Moodle

Es un proyecto *Open Source*, publicado bajo licencia *GNU*, y multiplataforma que puede ser instalado en cualquier ordenador que ejecute PHP y cualquier base de datos de tipo SQL. La licencia *GNU* permite que sea un proyecto global al que cualquier desarrollador puede contribuir. No obstante, la mayoría del desarrollo es realizado por *Moodle Pty Ltd* con sede en Australia.

Este proyecto se inició en 1999 aunque la arquitectura empleada actualmente surgió en 2001. La primera versión se lanzó en 2002 y la última el 8 de junio de 2010.[3] En la actualidad existen unos 50.000 sitios registrados que utilizan la plataforma. En España es utilizado por alrededor de 1300 centros educativos, en su mayoría institutos y universidades [24]

Desde su concepción se planteó como esencial la pedagogía colaborativa, siendo diseñado para que los alumnos pueden crear comentarios y contenidos en casi todas las herramientas.

aTutor

Es desarrollado por el *Adaptive Technology Resource Centre* de la Universidad de Toronto desde 2003[4]. La última versión se publicó a finales de 2009. Al igual que otros *LMS*, es un

[2] http://www.desire2learn.com/
[3] http://moodle.org/
[4] http://www.atutor.ca/

proyecto *Open Source*, publicado bajo licencia *GNU*, y multiplataforma. Prácticamente todo su código está escrito en PHP.

Destaca frente a otros *LMS* porque está diseñado para ser completamente accesible, desde cualquier medio, y herramienta (no sólo ratón y teclado) y por cualquier persona (incluso invidentes). Para ello, advierte de la necesidad de añadir texto alternativo al introducir imágenes y ofrece una interfaz adaptable, de manera que la forma en que los contenidos son presentados sea completamente personal. Cómo contrapartida, la interfaz pudiera ser considerada poco intuitiva, y difícil de utilizar mientras el usuario no se ha familiarizado con ella [23].

.LRN

.LRN[5] fue desarrollado por el *MIT* a partir de un *CMS* denominado *OpenACS*. Se trata de un proyecto de código abierto distribuido con licencia *GNU/GPL* y respaldado por el *LRN Consortium*, una asociación sin ánimo de lucro. Todas las nuevas versiones son revisadas por dicho consorcio [26].

Actualmente el número de usuarios de que dispone es considerablemente inferior al de algunos competidores, ya que tiene alrededor de medio millón de usuarios en el mundo, frente a los aproximadamente 10 millones de *Moodle* [22]. La principal ventaja de *.LRN* sobre ellos es que *OpenACS* utiliza el servidor *AOLServer*, el cual debido a su interprete de *TCL* integrado, resulta altamente escalable. Esto permite su uso en entornos de alta capacidad que soportan un número de usuarios muy elevado (como serían precisamente las universidades), sin perder rendimiento [19]. Uno de los motivos por los que ha ido perdiendo terreno respecto a sus competidores es la mejora del servidor *Apache*, utilizado por la mayoría de ellos. Las diferencias de ambos servidores en entornos grandes ya no resultan tan significativas.

Sakai

El desarrollo de *Sakai*[6] surge en 2004 tras una alianza de universidades y afiliados comerciales denominada *Sakai Community*, con la financiación de la fundación *Sakai Community*. A diferencia de otros proyectos libres que surgen sin presupuesto inicial y más tarde reciben donaciones, *Sakai* recibió una suma inicial de 2 millones de dólares proveniente de la *Mellon Foundation* [25].

[5] http://www.dotlrn.org/
[6] http://sakaiproject.org/

Sakai se distribuye con licencia de tipo *Open Source* denominada *Educational Community License*. La última versión fue lanzada en agosto de 2010. Actualmente es utilizado por unas 350 universidades e instituciones en todo el mundo.

2.1.2. Tendencias de los LMS

Con la evolución de la *web 2.0* surgen multitud de recursos *web 'en la nube'*, que ofrecen algunos de los servicios que requieren los *LMS*, tales cómo distribución de documentos, foros de discusión, *weblogs*,*bookmarking* y vídeos.

Frente a los *LMS* tradicionales ha surgido en los últimos años un modelo alternativo utilizado por parte de algunas instituciones y, sobre todo, por educadores individuales, consistente en integrar las herramientas mencionadas anteriormente en un *weblog* o plataforma similar [12] [14].

Este modelo tiene ofrece algunas ventajas frente al *LMS* tradicional. Las más reseñables son:

- Precio: La mayoría de los servicios se ofrecen de manera gratuita o a un coste muy reducido.
- Mantenimiento. Al encontrarse *'en la nube'* no es necesario disponer de un servidor para albergar los contenidos.
- Versatilidad: Es posible utilizar cualquier herramienta disponible para ser utilizada desde un navegador, no es necesario que este diseñada para un *LMS* en concreto. Esto permite también que sean los propios alumnos los que elijan nuevas herramientas.
- Portabilidad: Al alojar los contenidos fuera del *LMS*, los usuarios pueden seguir teniendo acceso a los contenidos que hayan publicado en la *web*.
- Rápida evolución: Las herramientas no están diseñadas únicamente para el aprendizaje, sino con un propósito general, y son desarrolladas por multitud de empresas de todos los ámbitos. Cómo consecuencia existe una innovación y mejora constante.

Sin embargo, este nuevo modelo alternativo presenta una serie de inconvenientes, entre los que podrían señalarse:

- Tiempo: La evaluación de las posibles herramientas alternativas precisa una dedicación por parte del personal docente o administrador, y por parte los alumnos requiere el aprendizaje de su uso.

- Mantenimiento: La prestación continuada del servicio depende de las empresas que lo ofrecen. Al ser prestado de manera gratuita en la mayoría de los casos, no existen garantías de que no se produzcan interrupciones del mismo.

- Seguridad: El control de los datos personales también recae en las empresas que ofrecen los servicios, por lo que las instituciones no pueden ejercer ninguna responsabilidad sobre ello.

- Continuidad: Al igual que ocurre con el mantenimiento, no existen garantías de que las herramientas vayan a seguir existiendo. En caso de que cesara su servicio, podría perderse no sólo la posibilidad de seguir utilizándolo, sino en algunos casos, también los contenidos que se han albergado en el mismo.

- Heterogeneidad: Al utilizar cualquier herramienta disponible en la *web*, no puede garantizarse que la oferta sea la misma para todos los usuarios. Algunas empresas solo ofrecen algunos servicios a determinados usuarios (por ejemplo, a los de un determinado país).

- Control: Así cómo dentro de un *LMS* es posible tener acceso a todo lo realizado por los alumnos, con este modelo no es posible conocer todo lo que realizan los usuarios.

- Autenticación: Aunque algunas herramientas ofrecen servicios de autenticación, generalmente estos no están sujetos a los protocolos de la institución.

La tendencia actual consiste en combinar lo mejor de los *LMS*s tradicionales con el nuevo modelo alternativo de utilización de herramientas *web* gratuitas. Es decir, crear *mash-ups* dentro de los *LMS*s tradicionales que permitan integrar las herramientas externas dentro del *LMS*, manteniendo la seguridad, control de acceso y autenticación [12]. Con esta integración no se consiguen todas las ventajas descritas anteriormente que ofrecen las herramientas externas, ni tampoco se eliminan todos los inconvenientes. Sin embargo, en conjunto podemos obtener ventajas que no obtendríamos individualmente en ninguno de los dos modelos.

2.2. Herramientas de cuestionarios online

Una de las funcionalidades que permiten las páginas *web* son los formularios. Se conoce cómo formulario *web* (o *webform*) a un elemento *HTML* que contiene una serie de campos en los que el usuario puede introducir información que puede ser enviada a otra página *web*.

Los *webform*s constituyen la principal herramienta de obtención de datos de los usuarios en las páginas *web*, utilizándose en numerosos ámbitos desde tiendas *online* a foros de discusión, en los que la *web* interpreta directamente los datos. En nuestro caso consideraremos únicamente los formularios utilizados cómo cuestionarios, es decir, aquellos en los que los datos introducidos por el usuario son registrados para la elaboración de encuestas o valoración de usuarios, lo cual resulta de utilidad en *eLearning* para la evaluación de los alumnos.

Los campos que puede tener un formulario pueden ser de los siguientes tipos:

- *text*: Cuadro de texto de una única línea.
- *textarea*: Cuadro de texto que permite más de una línea.
- *checkbox*: Casilla de selección (tiene dos estados: seleccionada y deseleccionada).
- *select*: Permite seleccionar un elemento de una lista.
- *file*: Permite subir un archivo.
- *radio*: Botón de selección. A diferencia de *checkbox*, los botones radio se muestran agrupados, siendo posible únicamente la selección de uno del grupo. Los botones de selección se utilizan en los cuestionarios para la realización de:
  - Elección entre varias opciones. En ocasiones acompañadas de una última opción 'otra' la cual acompaña a un cuadro de texto, en el que el usuario puede especificar otra opción que no se encuentra en la lista.
  - Escala de valores, en dónde cada uno de los botones asigna un valor cuantitativo.
  - Cuadricula o matriz, en el que cada fila es un conjunto de botones de selección a los que podemos asignar el valor de una única columna.

Con el auge de la *web 2.0* se han popularizado las herramientas online que permiten a los usuarios crear cuestionarios fácilmente, sin necesidad de tener conocimientos de *HTML* ni de disponer de un servidor en el que albergar tanto el *webform* cómo almacenar las respuestas. Además, muchas de estas herramientas ofrecen la posibilidad de analizar los datos, generando estadísticas y mostrándolas de manera gráfica.

Herramientas disponibles

Existen numerosas alternativas de herramientas para la realización de cuestionarios *online*. Debido a la necesidad de almacenar los datos de los resultados, la mayoría suelen tener una versión gratuita limitada, principalmente en el número de respuestas, y versiones comerciales con mayores funcionalidades.

De todas las herramientas existentes, se ha seleccionado un subconjunto representativo que se describe a continuación:

QuestionForm

La herramienta *QuestionForm* permite preguntas de tipo texto, test, *checkbox*, escala y cuadrícula o matriz. La versión básica es gratuita, pero es necesario pagar para acceder a la mayoría de las funcionalidades. Ofrece la posibilidad de integrar el cuestionario cómo código HTML embebido. Muestra estadísticas de los resultados y se pueden exportar a una hoja de cálculo.[7]

Crearcuestionarios

Únicamente permite preguntas de tipo test, lo cual implica que el volumen de datos generado por los formularios es reducido y por tanto es posible ofrecer el servicio de manera completamente gratuita. Las estadísticas se muestran públicamente y no es posible exportar los resultados.[8]

Free Online Surveys

Admite preguntas de todo tipo: texto, test, *checkbox*, escala, matriz y lista. A pesar de su nombre, sólo admite 20 preguntas por formulario y 50 respuestas durante 10 días en su versión gratuita. Es necesario pagar 30 dólares para poder ver las respuestas cuando éstas sean más de 50 o hayan pasado más de 10 días. Las estadísticas de los resultados son mostradas en distintas gráficas.[9]

Web Online Surveys

Permite preguntas de tipo texto, test y *checkbox*. Sólo permite de manera gratuita 25 respuestas, para que admita más es necesario pagar 20 dólares al mes. Los resultados se exportan

[7] http://www.questionform.com/
[8] http://www.crearcuestionarios.com/
[9] http://freeonlinesurveys.com

para ser utilizados en una hoja de cálculo.[10]

Encuestafacil

Admite preguntas de texto, test, *checkbox*, escala y matriz. Además de mostrar el cuestionario cómo código embebido o en su página *web*, dispone de la posibilidad de imprimir los cuestionarios en papel y posteriormente reconocer las respuestas seleccionadas tras digitalizar el papel en que se encuentran las respuestas. Existe una versión básica gratuita y varias comerciales. Elabora análisis de los resultados, los cuales sólo están disponibles para el creador.[11]

Surveymonkey

Permite preguntas de todo tipo (texto, test, *checkbox*, escala, matriz y lista). Admite la posibilidad de crear lógica de exclusión, es decir, en función de la respuesta a una pregunta, modificar las preguntas posteriores. Ofrece seguridad en el envío de respuestas mediante *SSL*, lo cual garantiza la confidencialidad. Muestra los resultados en distintas gráficas y permite su exportación en formato *Excel*, *CSV* y *XML*. Además, es posible compartir y mostrar de manera pública a los usuarios un resumen de las respuestas recibidas, aplicando previamente filtros para mostrar únicamente una parte de la información.

La versión gratuita sólo permite la utilización de algunas funciones y un máximo de 100 respuestas. Existe una versión limitada mensual de 19,95 euros y una versión completa por 200 euros.[12]

Google Formularios

Dentro del paquete de herramientas *online* denominado *Google Docs*[13] disponemos de una herramienta de formularios. Esta herramienta permite crear cuestionarios con todo tipo de preguntas, con la opción de éstas tengan lógica de exclusión. También ofrece la posibilidad de mostrar de manera gráfica a todos los realizadores del cuestionario las estadísticas de las respuestas obtenidas.

Google Formularios almacena las respuestas en una hoja de cálculo de la herramienta del

[10] http://web-online-surveys.com/
[11] http://www.encuestafacil.com/
[12] http://es.surveymonkey.com/
[13] http://docs.google.com/

mismo paquete denominada *Google Spreadsheets*. El hecho de disponer de las respuestas en una hoja de cálculo nos permite trabajar directamente en la realización de operaciones con las mismas. Las hojas de cálculo de *Google Spreadsheets* se pueden compartir con otros usuarios, y además pueden importar datos en formato *Excel* (.xls y .xlsx), *OpenOffice* (.ods), *CSV* y texto (.txt); y exportar en formato *CSV*, *HTML*, texto (*.txt*), *Excel* (sólo *.xls*), *OpenOffice* (*.ods*) y *PDF*.

Además, es posible tener acceso a dicha hoja de cálculo de *Google Spreadsheets* a través de un *API* pública, mediante comandos *HTTP* y *XML*.

Capítulo 3

Entorno de desarrollo

En este capítulo se analizan y describen las tecnologías que han sido utilizadas en la realización de este proyecto, así como las razones que han llevado a elegir el uso de algunas herramientas en aquellos casos en los que existen otras alternativas.

3.1. Tecnologías de .LRN

En esta sección se detallan todas las tecnologías necesarias u opcionales para el funcionamiento de la plataforma *.LRN*.

3.1.1. OpenACS

Open Architecture Community System (*OpenACS*) es un *framework* para aplicaciones *web* basadas en contenido dinámico dentro del marco de comunidades de usuarios. Se trata de un proyecto *OpenSource* distribuido mediante licencia *GNU/GPL*. En 2002 ante la decisión de portar el código de *ACS* a Java, surgió *OpenACS*, el cual continua utilizando código *TCL*, siendo perfeccionado por una comunidad de desarrolladores. Las decisiones técnicas son llevadas a cabo por el *OpenACS Core Team* (*OCT*), el cual es elegido periódicamente en votación. *OpenACS* se ejecuta sobre un servidor *AOLServer* y puede utilizar como base de datos *PostgreSQL* u *Oracle*.

La arquitectura de *OpenACS* ofrece tanto una serie de aplicaciones de tipo colaborativo (foro, *chat*, etcétera), como un kit de desarrollo que permite la fácil realización de nuevas aplicaciones (para lo cual el desarrollador dispone de un modelo de objetos, permisos y plantillas.). La estructura de su arquitectura puede verse, como parte de *.LRN* en la figura 3.1.

3.1.2. .LRN

.LRN (pronunciado 'dot learn') es un Entorno Virtual de Aprendizaje (*LMS*) desarrollado como especialización del *framework OpenACS* (descrito en la sección anterior, 3.1.1), aprovechando su utilidad como Sistema de Gestión de Contenidos (*CMS*). Cómo se ha comentado en el capítulo anterior, la parte fundamental de un *LMS* puede ser la especialización de un *CMS* para el ámbito de *eLearning*.

Básicamente, *.LRN* añade una capa a *OpenACS* compuesta por una serie de aplicaciones que utilizan la infraestructura de *OpenACS* y un portal que da acceso a las mismas.

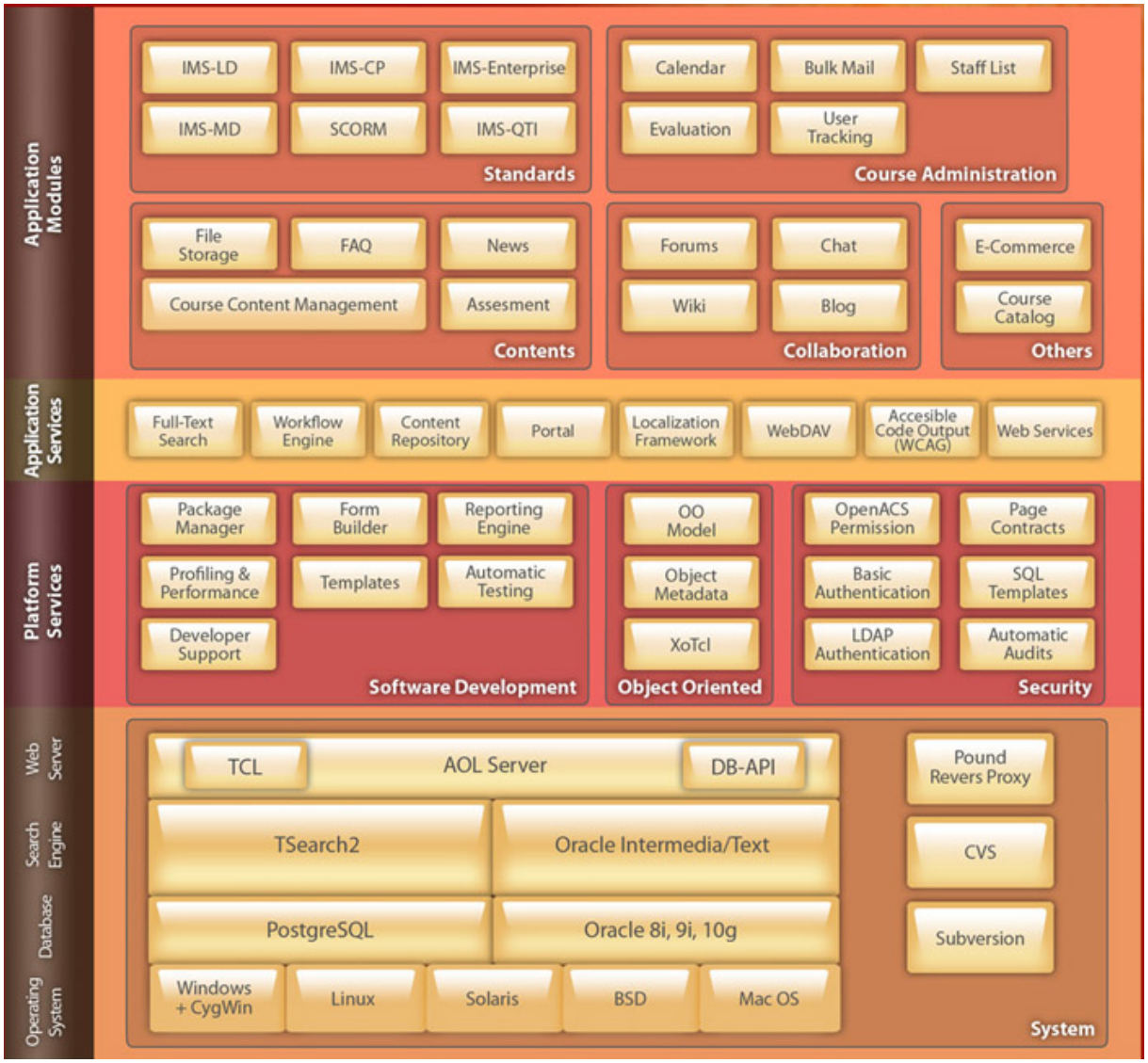


Figura 3.1: Esquema de *.LRN*. (Fuente: http://www.dotlrn.org)

El hecho de estar basado en *OpenACS* permite que sea altamente escalable, pudiendo admitir gran cantidad tanto de usuarios como de recursos. Los usuarios y administradores pueden definir distintos tipos de comunidades, cada una de las cuales puede integrar sus propias herramientas y recursos.

Cualquier desarrollador puede realizar una nueva aplicación de *.LRN*. Para que esta forme parte del proyecto *.LRN*, en una primera fase es publicada como *'.LRN compatible'*. Posteriormente, en una segunda fase, es llevado a cabo un proceso de verificación de la misma por parte del Consorcio *.LRN*, en el que se comprueba su estabilidad y tras ello es publicada como *'.LRN Certified'*.

Las aplicaciones certificadas que componen la última versión de *.LRN* son las siguientes: foros, almacenamiento de archivos, calendario, noticias, encuestas, *FAQ* y listas de correo. Otros módulos disponibles pero todavía no certificados son: *blogs*, autoevaluaciones, agregador de noticias *RSS*, editor *HTML*, *Powerpoint web*, encuestas complejas y álbum de fotos.

Al igual que *OpenACS*, toda la arquitectura completa de *.LRN* está basada en herramientas *OpenSource*. El diagrama detallado de la arquitectura se muestra en la figura 3.1.

3.1.3. AOLserver

AOLserver es el servidor *web* multiplataforma utilizado por *OpenACS*. El código de *AOLserver* está liberado desde 1999, cuando *America Online* decidió hacerlo [20]. Desde entonces se distribuye con una licencia denominada *AOLserver Public License*, la cual es similar a la de *Mozilla*.

Está especialmente diseñado para el uso en *webs* de contenidos dinámicos, ya que mantiene de manera permanente un fondo de conexiones con la base de datos, de manera que no es necesario abrir una nueva conexión cada vez que se requiere cargar una página.

Aunque también soporta páginas estáticas, está diseñado para la utilización de páginas de contenidos dinámicos (*ADPs*) que combinan *HTML* con código *TCL*. Dicho código es ejecutado en el servidor de manera eficiente debido a un procesamiento multihilo.

3.1.4. AOLserver Dinamic Pages (ADPs)

Las páginas *ADPs* son páginas *HTML* que incluyen contenido dinámico procedente de la ejecución de *scripts TCL*. Cada vez que se accede a una página *ADP*, si existe un *script TCL*

asociado con dicha página, se ejecuta previamente éste. A continuación se ejecutará la página *ADP*, utilizando el contenido generado en el *script* anterior.

Las páginas *ADPs* pueden contener tanto etiquetas *HTML* (tags) como otras propias que constituyen el código necesario para generar la página. Este código incluye bucles, condicionales y *scripts TCL* embebidos. Además, el contenido de las páginas *ADPs* es reutilizable, pudiéndose emplear el código de unas de ellas en otras, con una simple llamada.

Por esta razón, las páginas *ADPs* son útiles tanto para el acceso e inserción de elementos en una base de datos (utilizando para ello los *scripts* definidos en el fichero *TCL*), como para la generación de código *HTML* de manera condicional. [1]

3.1.5. Tool Command Language (Tcl)

Tcl (pronunciado */tí.quel/*) es un lenguaje interpretado, concebido por su creador como un lenguaje de sintaxis sencilla que permite un fácil aprendizaje, pero sin tener que renunciar a la funcionalidad. El código *Tcl* es generalmente más compacto y legible que el de otros lenguajes de programación. Actualmente el desarrollo corre a cargo del *Tcl Core Team* [4]. Es un lenguaje multiplataforma, al existir interpretes para casi todos los sistemas operativos.

Se utiliza principalmente para la realización de interfaces gráficas, aunque también para el desarrollo de prototipos o aplicaciones de tipo *'script'*, como es el caso del servidor *AOLserver*.

3.1.6. PostgreSQL (PSQL)

OpenACS soporta bases de datos *Oracle* y *PostgreSQL*. Se ha utilizado como base de datos *PostgreSQL* por ser *software* libre, a diferencia de *Oracle*, que tiene licencia privativa.

Al igual que otro proyectos *OpenSource*, *PostgreSQL* es desarrollado por una comunidad de usuarios, denominada en su caso *PGDG* (*PostgreSQL Global Development Group*). *PSQL* es multiplataforma y se publica bajo licencia *BSD*, una licencia más permisiva que *GPL* al permitir la utilización del *software* en productos comerciales.

3.1.7. Transport Layer Security (TLS/SSL)

TLS es un protocolo criptográfico, sucesor de *SSL*, que proporciona comunicaciones encriptadas a través de una red de datos, como internet, a nivel de aplicación.

OpenACS soporta *TLS* de manera opcional. Para ello, es necesario instalar una extensión de *OpenSSL* para *TCL*[1]. Dicho paquete es *OpenSource*.

Se ha utilizado para otorgar una mayor seguridad a las datos transferidos entre la aplicación y *Google*, siendo un requisito para la aplicación la instalación de *TLS* (ya que también es un requisito para utilizar el *API* de *Google Docs*).

Para que las conexiones entre el cliente y el servidor también sean seguras y utilicen *TLS*, es necesaria su pertinente configuración en el servidor, lo cual es ajeno a la aplicación.

3.1.8. TclCurl

A pesar de que *OpenACS* admite conexiones seguras (*TLS/SSL*) con los clientes, los métodos que incluye su *API* para establecer conexiones *HTTP* con otros servidores, no utilizan este tipo de conexiones (*HTTPS*). Por ello, ha sido necesario utilizar la librería *TclCurl*[2] para el desarrollo de la aplicación.

La librería *TclCurl* es una adaptación desarrollada por Andrés García a partir de la librería *libCurl*, la cual fue desarrollada principalmente por Daniel Stenberg. Esta librería es multiplataforma y se publica bajo la licencia *BSD* (la misma que es utilizada por *PSQL*).

TclCurl permite hacer todo tipo de peticiones *HTTP*, requiriendo *OpenSSL* para todas aquellas conexiones que precisan ser seguras (*HTTPS*).

3.1.9. OpenACS Template System (ATS)

El sistema de plantillas de *OpenACS* (*ATS*)[3] ha sido diseñado para permitir a los desarrolladores separar claramente la 'lógica de aplicación' (en un fichero *TCL*) de la 'lógica de diseño' (en un fichero *ADP*). Esto ofrece la posibilidad de que desarrolladores y diseñadores gráficos puedan trabajar de manera independiente.

Cómo se ha explicado anteriormente, por cada página *ADP* normalmente existe su correspondiente *script TCL*. Dicho *script TCL* se ejecuta en primer lugar, y a continuación se procede al análisis de la página *ADP*. El *ATS* utiliza las variables definidas en el *script TCL* (denominadas *data source* o *page property*) para generar código *HTML*, en función de la etiqueta especificada en el fichero *ADP* y del valor de dichas variables.

[1] http://www.openacs.org/doc/install-ssl.html
[2] http://personal5.iddeo.es/andresgarci/tclcurl/english/index.html
[3] http://openacs.org/doc/templates.html

Las etiquetas que han sido utilizadas para realizar la aplicación son *formtemplate* para obtener los datos suministrados por el usuario (a la hora de crear y editar cuestionarios) y *listtemplate* para mostrar los datos en el resto de páginas (lista de cuestionarios disponibles y lista de respuestas).

3.1.10. ACS Content Repository

El *Content Repository*[4] es una extensión del modelo de objetos proporcionado por *OpenACS* (*ACS Object Model*). Está diseñado para almacenar cualquier tipo de 'contenido' en la base de datos. Sus principales ventajas son: soporte de versiones, organización jerárquica de elementos, sistema de permisos y extensión de tablas de manera dinámica.

La utilización del *Content Repository* requiere manejar los objetos de las tablas que hacen uso de él de manera especial, lo cual supone un importante proceso de aprendizaje, sin embargo simplifica bastante el proceso de creación de nuevas tablas, ya que el contenido de éstas suele ser homogéneo y en la mayoría de los casos requiere características que aporta el *Content Repository*.

En el caso de la aplicación desarrollada, se ha utilizado el soporte de versiones para la edición de parámetros de los formularios. La organización jerárquica se ha utilizado para la relación entre la tabla de formularios y la de respuestas. Además, se han utilizado sus funciones para crear (al instalar la aplicación) y eliminar (al desinstalar) las tablas de la base de datos, encargándose de generar automáticamente todos los permisos correspondientes.

En la figura 3.2 se muestra la relación entre los distintos tipos de objetos del *Content Repository*. Para desarrollar la aplicación se han creado nuevos tipos (*Custom Content Types*) para cada tabla. Por cada objeto de dichas tablas es necesario que exista su correspondiente objeto *Content Revision* y *ACS Object*, y además, otro objeto de *Content Item*, el cual también se corresponde unívocamente con otro *ACS Object*. Este ítem puede ser compartido por varios objetos de los anteriores, si estos fueran distintas versiones de un mismo elemento (en el caso de los formularios, si sus atributos han sido editados), siendo únicamente uno el actual (*live*). Todo esto se explicará con más detalle en el apartado 4.3 del siguiente capítulo.

[4] http://openacs.org/doc/acs-content-repository/

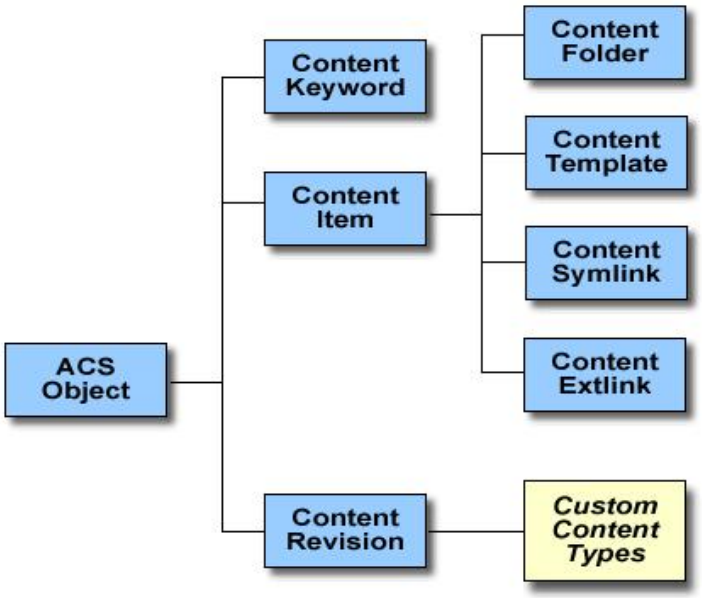


Figura 3.2: Esquema de objetos del *Content Repository* (Fuente: http://www.openacs.org)

3.1.11. tDOM

Disponemos principalmente dos paquetes para el procesado de documentos *XML* en *TCL*: *TCLXML* y *tDOM*. *TCLXML* está escrito puramente en *TCL* y consta de tres paquetes distintos (*TclXML for parsing*, *TclDOM* y *TclXSLT*). En cambio, *tDOM* es un único paquete y no está escrito por completo en *TCL*, ya que parte de su implementación es en *C*). Para el desarrollo de la aplicación, se ha utilizado el paquete *tDOM* por estar incluido en la instalación de *OpenACS*.

tDOM es un paquete que utiliza muy poca memoria y es muy potente [5]. Tiene gran cantidad de funciones, entre las que resulta imprescindible para la aplicación que hemos desarrollado, la posibilidad de analizar sintácticamente (*parse*) código *HTML* que no cumpla todas las reglas de *XML* (como terminar con /> los elementos vacíos).

El análisis sintáctico (*parse*) de *XML* es empleado en nuestra aplicación tanto para la modificación de formularios *HTML* como para el uso del *API* de *Google Docs*.

3.2. Tecnologías de Google Docs

Google Docs está constituido por un conjunto de programas ofimáticos basados en *web*. Es un servicio gratuito que incluye las siguientes aplicaciones:

- Documentos (*Google Documents*)
- Hojas de cálculo (*Google Spreadsheets*)
- Presentaciones (*Google Presentations*)
- Dibujos (*Google Drawings*)
- Formularios (*Google Forms*)

Google Docs no sólo permite importar y exportar los archivos utilizados en una gran variedad de formatos, sino también compartirlos fácilmente con otros usuarios, con permisos completos (modificación) o sólo lectura. Del mismo modo puede ser utilizado para almacenar y compartir archivos de cualquier tipo de formato (incluidos aquellos que no se pueden manipular con las aplicaciones de *Google Docs*.

3.2.1. Google Forms

A diferencia de otras aplicaciones de *Google*, actualmente no existe un *API* para la utilización de *Google Forms* por parte de aplicaciones externas. Antes de la versión 3 de *Google Docs* era posible obtener mediante el *API* la lista de Formularios de un usuario, pero actualmente no está disponible dicha funcionalidad, por lo que ha de obtenerse de otra manera que se explicará tanto a lo largo de esta sección, como en la sección 4.5.2 del siguiente capítulo.

Sin embargo, *Google Forms* es una aplicación que en realidad es una extensión de *Google Spreadsheets*, la cual tiene un *API* pública disponible. Todos los formularios tienen asociada una Hoja de Cálculo (*Spreadsheet*) y cada una de ellas puede tener uno o ningún formulario asociado, pero no es posible tener más de un formulario en una misma *Spreadsheet*.

La aplicación se ha desarrollado teniendo en cuenta esta relación de *Google Forms* con *Google Spreadsheets*, las posibilidades que ofrece el *API* de *Google Spreadsheets* y las posibilidades que ofrece *Google Forms*, analizando su potencial de utilización mediante su interfaz de usuario, así como la capacidad de interacción mediante peticiones *HTTP* que simulen las que realizaría el usuario con el navegador.

A lo largo de las siguientes secciones se analiza el funcionamiento de *Google Forms*.

Creación de formularios

Los Formularios pueden ser creados sobre una *Spreadsheet* ya existente, o directamente.

En cualquier caso, el usuario tiene que tener iniciada sesión en su cuenta de Google. Si se crea el formulario directamente, el propietario será dicho usuario, mientras que si se crea a partir de una *Spreadsheet* el usuario tiene que ser el propietario o tener acceso de modificación (en caso de estar compartida).

- Creación de un formulario sin *Spreadsheet* previa.

  Para crear un formulario directamente, el usuario selecciona *'Formulario'* en el menú *'Crear nuevo'* de la página `http://docs.google.com/`.

  Esto genera una petición a `http://spreadsheets.google.com/newform`.

  Tras guardar el formulario se crea una *Spreadsheet* asociada al formulario con una columna inicial *'Timestamp'*, seguida por las columnas con los campos que se hayan creado en el formulario.

- Creación de un formulario a partir de una *Spreadsheet* ya existente.

  Para crear un formulario a partir de una *Spreadsheet*, el usuario selecciona *'Crear un formulario'* en el menú *'Formulario'* de la *Spreadsheet*, la cual está ubicada en la página `http://spreadsheets.google.com/ccc?key=claveSpreadsheet`.

  Esto genera una petición a `http://spreadsheets.google.com/gform?key=claveSpreadsheet`

  Se ha denominado como `claveSpreadsheet` a la clave que identifica a la *Spreadsheet*.

Claves de *Spreadsheet* y Formulario

A pesar de la relación unívoca existente de cada Formulario con una única *Spreadsheet* (y viceversa), existen dos claves distintas utilizadas para identificar a Formularios y *Spreadsheet*s.

Se ha realizado una comparación entre dichas claves para averiguar la relación existente entre cada clave de Formulario con su correspondiente clave de *Spreadsheet* y se ha descubierto que entre ambas existen 31 caracteres comunes, a los que se añaden otros caracteres. Estos caracteres pueden ser sustituidos por otros de forma que a partir de una clave podemos generar una clave

válida para el otro caso. En el caso de la clave de *Spreadsheet* esto no será suficiente para acceder a las páginas que la utilizan, ya que estas requieren tener iniciada la sesión de usuario en *Google* y tener permisos para acceder a dicha *Spreadsheet*.

- Clave de *Spreadsheet* (`key`)

  La clave de *Spreadsheet* se utiliza para todas aquellas acciones que requieren tener iniciada sesión en la cuenta de Google y ser el propietario o tener acceso. Por ello, también se utiliza para modificar el Formulario. Se podría considerar por tanto una clave privada.

  Esta clave se denomina `key` y consta de 44 caracteres, los cuales se han comprobado que tienen siempre la siguiente estructura:

  - 2 caracteres siempre iguales: `0A`
  - 11 caracteres que identifican al usuario propietario
  - 31 caracteres que identifican al conjunto Formulario y *Spreadsheet* (los mismos que para la `formkey` del Formulario)

  Se ha comprobado que los 11 caracteres que identifican al usuario propietario no son necesarios para acceder a la *Spreadsheet* (sí lo es siempre tener iniciada una sesión Google que tenga acceso), siendo posible el acceso cuando se sustituyen por 11 caracteres cualesquiera.

- Clave de Formulario (`formkey`)

  La clave de Formulario se utiliza en las acciones de acceso al mismo para su cumplimentación. Es decir, acciones que no requieren tener iniciada una sesión de *Google*, y son accesibles por todo aquél que conozca esta clave. Se podría considerar por tanto una clave pública, ya que es divulgada al publicar la *URL* del Formulario.

  Esta clave se denomina `formkey` y consta de 34 caracteres que se ha comprobado que siguen la siguiente estructura:

  - 31 caracteres que identifican al conjunto Formulario y *Spreadsheet* (los mismos que para la `key` de la *Spreadsheet*)
  - 3 caracteres adicionales que pueden ser `6MA` ó `6MQ`
  - Por último, al final de la clave pueden aparecer ninguno, uno, o dos puntos.

El procedimiento que se ha seguido para llegar a la conclusión de que los 3 últimos caracteres son siempre `6MA` ó `6MQ` ha consistido en buscar *URLs* públicas de formularios utilizando el buscador de *Google* y observar que de las 50 primeras *URLs* encontradas, todas terminan con dichos caracteres. Se cree que utilizan `6MA` los formularios más antiguos y `6MQ` los más recientes.

Se ha comprobado que las peticiones son llevadas a cabo correctamente existan o no los últimos puntos. También funcionan correctamente sustituyendo el último carácter por cualquier otro, así como la letra del penúltimo carácter `M` por la `N`.

Es decir, podemos acceder al Formulario utilizando:

- 31 caracteres que identifican al conjunto Formulario y *Spreadsheet*
- 3 caracteres adicionales:
  - Carácter `6`
  - Carácter `M` ó `N`
  - Cualquier carácter
- Ninguno, uno, o dos puntos.

Podemos obtener las claves (`key`) de cada *Spreadsheet* a la cual tiene acceso un usuario, utilizando el *API* de *Google Spreadsheets*. Por tanto, podemos conocer los 31 caracteres del conjunto Formulario y *Spreadsheet*, y a partir de éstos, crear una clave válida para el Formulario asociado a dicha *Spreadsheet*, añadiendo a los 31 caracteres cualquier sufijo válido, aunque no sea el original del formulario.

*URL*s utilizadas por la interfaz de usuario

Se ha observado cuales son las *URL*s utilizadas por la interfaz de usuario de *Google Docs* para realizar operaciones desde el navegador, con la finalidad de poder utilizar dichas *URL*s para realizar peticiones desde la aplicación.

La identificación del Formulario (o *Spreadsheet*) se hace mediante el valor del parámetro `key` o `formkey` (en función de si el acceso es privado o público). Además del parámetro de la clave, pueden tener otros parámetros adicionales como el idioma (`hl`).

Podemos clasificar estas *URLs* en función de la clave que utilicen, de la siguiente manera:

- *URL* sin clave

  - `https://spreadsheets.google.com/newform` Crea un nuevo formulario y redirige a la URL de modificación de formulario, para lo cual requiere tener iniciada una sesión de *Google*.

    Esta *URL* se utiliza en la aplicación para proporcionarle al usuario desde la aplicación un enlace a la página de creación de cuestionarios.

    Debido a que la *URL* redirecciona automáticamente y ante la dificultad de analizar el código *HTML* de la interfaz de *Google Docs*, esta *URL* ha sido descubierta gracias al uso de la herramienta *Wireshark*, cuyo funcionamiento se explicará más adelante en la sección 3.3.5.

- *URL*s de acceso mediante la clave de *Spreadsheet* `key` (requieren correspondiente sesión *Google*)

  - `https://spreadsheets.google.com/gform?key=claveSpreadsheet`

    Esta *URL* muestra la interfaz de modificación de formulario. En dicha interfaz el usuario asigna los campos, valores y diseño que tendrá la *Spreadsheet*.

    Si es la primera vez que se crea el formulario, a partir de una *Spreadsheet* ya existente, se añadirá una columna denominada 'Timestamp' y a continuación las columnas de los campos del formulario.

    Si ya existiera una columna de 'Timestamp' creada por un formulario anterior que se hubiese eliminado, al crear de nuevo un formulario dicha columna se mantendría, añadiéndose una nueva columna 'Timestamp' que utilizará el nuevo formulario.

    Además, como se explicará en la sección de *Google Spreadsheets*, se modificará una tabla de la *Spreadsheet*.

  - `https://spreadsheets.google.com/ccc?key=claveSpreadsheet`

    Esta *URL* muestra la *Spreadsheet*, sobre la cual el usuario puede realizar modificaciones.

    Existen ciertas restricciones en las modificaciones que el usuario puede realizar cuando dicha *Spreadsheet* tiene un Formulario, por la existencia de la tabla correspondiente a dicho formulario, como se explicará en la sección 3.2.2.

- *URL*s de acceso mediante la clave de formulario `formkey` (accesibles de manera pública por cualquiera que conozca la clave)

  - https://spreadsheets.google.com/viewform?formkey=claveFormulario

    Esta es la *URL* pública del formulario. El formulario que contiene envía los datos a la *URL* de `formResponse`.

  - https://spreadsheets.google.com/embeddedform?formkey=claveFormulario

    Igual que en el caso anterior, es una *URL* pública del formulario, con la diferencia de que está destinada a su utilización dentro de marcos (etiqueta de *HTML* `iframe`), de manera que el formulario pueda ser embebido en páginas *web*.

    Una diferencia respecto al anterior es que este no dispone de todas las Hojas de Estilo (*CSS*) que le dan un aspecto más amigable al formulario. Sí posee una única Hoja de Estilo encargada de la distribución y correcta visualización de los elementos del formulario.

    Al igual que en el caso anterior, los datos del formulario son enviados mediante una petición a la *URL* de `formResponse` pero con la diferencia de que es añadido el campo `embedded=true`.

  - https://spreadsheets.google.com/formResponse?formkey=claveFormulario

    A esta *URL* llegan los datos del formulario. Aunque los formularios de las *URLs* anteriores son creados para enviar los datos mediante el método `POST`, se ha comprobado que esta *URL* funciona igualmente si se envían mediante el método `GET`.

    A la hora de crear un formulario se pueden marcar ciertos campos como obligatorios. En dicho caso, es posible que no se hayan cumplimentado. De ser así, al no recibirse los valores de dichos campos en esta *URL*, lo que se mostraría es de nuevo el formulario, de manera similar a cómo lo hacen las *URL* anteriores (embebida o no), señalando cuales son los campos que no se han rellenado, y manteniendo los valores de aquellos que sí se rellenaron (y que por tanto han sido recibidos en esta *URL*).

    En el caso de que se reciban valores para todos los campos requeridos (o no existan campos obligatorios), estos son añadidos a la *Spreadsheet* y es mostrado un mensaje personalizable que además puede contener un enlace a la siguiente *URL*.

  - https://spreadsheets.google.com/viewanalytics?formkey=claveFormulario

Esta página muestra un resumen de todas las respuestas recibidas, siempre y cuando haya sido seleccionada la opción 'Publicar resumen de respuestas' en la interfaz de creación y modificación del formulario.

En el caso de las preguntas que son campos de texto, son mostradas todas las respuestas recibidas, mientras que para el resto de tipo de preguntas se representa mediante un gráfico el número de respuestas que se han recibido de cada opción.

Esta *URL* se ha utilizado para proporcionarle al usuario desde la aplicación un enlace al resumen de las respuestas del cuestionario.

El código *HTML* de esta página está basado en *javascript*, lo cual dificulta el análisis de la página completa. Realizando una petición `GET` a dicha *URL* no se obtiene la página completa, ya que faltan los datos de las respuestas.

Utilizando la herramienta *Wireshark* se ha comprobado que cuando se accede a dicha *URL* utilizando un navegador, éste realiza una petición de tipo `POST` incluyendo en el cuerpo de la petición `timestamp=0&token&id` a la *URL* que viene a continuación.

- `https://spreadsheets.google.com/analytics?formkey=claveFormulario`

  Como contestación a dicha petición se reciben las respuestas de los formularios en formato *javascript*.

3.2.2. Google Spreadsheets

Google Spreadsheets dispone de un servicio de *API* [5] que nos permite acceder y modificar el contenido de las *Spreadsheet*s.

Para acceder a dicho servicio primero es necesario autenticar al usuario, utilizando los procedimientos que se describirán en la sección 3.2.3. Una vez autenticado el usuario podemos realizar distintas peticiones, tales como obtener la lista de todas sus *Spreadsheet*s (*'metafeed'*), lo cual proporciona la clave de identificación (`key`) de dichas *Spreadsheet*s, entre otra información como *email* de los propietarios y fecha de última modificación.

Disponiendo de la `key` de una *Spreadsheet*, podemos realizar peticiones para obtener las siguientes fuentes (*'feeds'*):

- *Worksheet* (Hoja). Aunque los resultados de los formularios aparecen en la primera *Worksheet*, una *Spreadsheet* puede estar formada por hasta 100 *Worksheet*, cada una con sus

[5] `http://code.google.com/intl/es-ES/apis/spreadsheets/data/3.0/developers_guide.html`

distintas filas y columnas. Realizando una petición de las fuentes (*'feeds'*) de *Worksheets* de una *Spreeadsheet* obtenemos los identificadores de las *Worksheets* existentes.

Una vez disponemos del identificador *'worksheetId'* de la Worksheet (además de la clave (`key`) de la *Spreadsheet*), podemos trabajar con los siguientes *'feeds'*:

- *List* (Lista). Agrupa los datos de la *Worksheet* por filas, y teniendo como identificador de cada columna el valor que toma la primera fila.
- *Cell* (Celda). Trata cada una de las celdas de la *Worksheet* de manera independiente.

- *Table* (Tabla). Una tabla es un conjunto de datos organizados que permite realizar ciertas acciones con el fin de que la información sea coherente y permanezca estructurada[6]. Cuando se introducen respuestas de formulario, estas se almacenan en una tabla como registros (*'records'*) independientes. Al crear o modificar la estructura de un formulario, se crea o modifica dicha tabla. Esto añade a la primera fila los nombres de las preguntas del formulario, precedidos por una columna denominada *'Timestamp'*, en la que se genera automáticamente la fecha y hora en la que es recibido cada formulario. Las tablas imposibilitan unir celdas, enlazar respuestas y modificar la primera fila de cada columna.

  Solamente es posible crear y modificar tablas utilizando el *API* de *Google Spreadsheets* y utilizando Formularios. La interfaz de usuario de *Google Spreadsheets* no permite la creación ni modificación de tablas.

  Realizando una petición de *'feeds'* de tablas se obtiene información sobre las tablas existentes, su identificador y la estructura de cada tabla. Conociendo el número de la tabla podemos obtener el siguiente *'feed'*:

  - *Records* (Registros). Nos muestran los valores que toman cada uno de los registros de la tabla. Para el caso de los formularios, son cada una de las respuestas que se han recibido.

El hecho de que los formularios creen tablas y se almacenen los resultados como registros simplifica el tratamiento por parte de la aplicación de los datos de la *Spreadsheet*. Sin embargo, se ha observado que los Formularios más antiguos, probablemente aquellos que tienen la `formkey` terminada en `6MA`, no utilizan tablas. Esto supone una limitación de la aplicación, ya que no podrá

[6] http://docs.google.com/support/bin/answer.py?answer=156100&hl=es

funcionar con formularios creados con gran anterioridad. Sería posible el tratamiento de los datos de la *Spreadsheet* accediendo por *'feeds'* de tipo lista, pero se ha considerado que la compatibilidad con dichos formularios no aporta suficiente valor a la aplicación como para justificar un diseño más complejo y en consecuencia menos fiable.

3.2.3. Autenticación para el uso del API de Google

Existen tres protocolos con los que es posible que una aplicación *web* realice la autenticación del usuario en el servidor de *Google* y así poder manejar los datos de *Google Docs*[7]:

- OAuth
- AuthSub
- ClientLogin

OAuth es un protocolo abierto diseñado para que, mediante un *API*, un servicio pueda acceder a los datos de un usuario en otro servicio. *OAuth* puede combinarse con *OpenID*, un estándar de identificación digital descentralizado, que permite a un usuario identificarse en una *web* y verificar dicha identificación en cualquier otro servidor que soporte el protocolo.

Aunque Google recomienda la utilización del protocolo *OAuth* siempre que sea posible, se ha considerado demasiado compleja su implementación para el tipo de aplicación desarrollada. Todas las peticiones de *OAuth* necesitan ir firmadas, por lo que es necesario realizar el registro del dominio que utiliza la aplicación y 'subir' (*upload*) dicho certificado público a *Google*. Esto significa que de haberse implementado este protocolo, cada servidor de *.LRN* que quisiera utilizar la aplicación necesitaría realizar dicho proceso, y proporcionar a la aplicación la clave privada con la que realizar dichas firmas. Se ha decidido no implementar *OAuth* al considerar que no es tan importante el nivel de seguridad que aporta el firmar todas las peticiones utilizando *AuthSub*, ya que éstas son realizadas utilizando *TLS/SSL*, y que es preferible que sea más sencilla la instalación de la aplicación, de manera que no sea necesario registrar el dominio en *Google*.

AuthSub en cambio no requiere registrar la aplicación en Google, aunque lo permite de manera opcional. Sí es necesario para acceder a algunos servicios, y para modificar e incluso suprimir la página en la que el usuario tiene que permitir a la aplicación acceder y modificar sus datos. El servicio de *Google Spreadsheets* no requiere registrar la aplicación para utilizar su *API*, así que

[7] http://code.google.com/intl/es-ES/apis/accounts/docs/AuthForWebApps.html

se ha podido implementar el proceso de autenticación utilizando *AuthSub* sin llevar a cabo dicho registro.

ClientLogin ha sido descartado por no aportar suficiente seguridad desde el punto de vista del usuario. En *ClientLogin* el usuario debe proporcionar a la aplicación el *'login'* y *'password'* que utiliza para acceder a su cuenta *Google*. Por tanto, la aplicación *web* puede almacenar esos datos que tan importantes que sirven para acceder a cualquier servicio de *Google* (incluyendo el correo electrónico de *GMail*). Por este motivo, *ClientLogin* no está diseñado para utilizarse en aplicaciones *web*, sino en aquellas aplicaciones instaladas en el ordenador del usuario, el cual no debería proporcionar nunca a las aplicaciones *web* sus datos de *login*. De implementar en una aplicación *web* como la nuestra dicho protocolo, algunos usuarios no querrían utilizarla por no confiar en el tratamiento que esta pueda realizar con sus datos.

A diferencia de lo que ocurre con *ClientLogin*, con *AuthSub* el cliente proporciona su *'login'* y *'password'* no a la propia aplicación, sino al servidor *Google*, pudiendo comprobar en su navegador que introduce los datos en una página del dominio de *Google*. En dicha página, el usuario tiene que autorizar la utilización de los datos de su cuenta pertenecientes a un servicio (en este caso *Google Spreadsheets*), no pudiendo acceder la aplicación a ningún dato de la cuenta de *Google* que sea de otro servicio. El usuario puede además en todo momento revocar dicha autorización.

Por los motivos que han sido expuestos, *AuthSub* ha sido el protocolo de autenticación elegido para implementar en nuestra aplicación. En la sección 4.5.1 se detallará cómo se lleva a cabo dicho proceso de autenticación.

3.3. Otras herramientas utilizadas

A continuación se detallan todas aquellas herramientas que han sido utilizadas para la realización del proyecto pero que no forman parte ni de *.LRN* ni de *Google Docs*.

3.3.1. VMware

Con el objetivo de desarrollar la aplicación de una manera portable y sin necesidad de tener que instalar una plataforma de *.LRN*, se ha utilizado una máquina virtual creada por la Universidad Carlos III de Madrid[8]. Dicha máquina virtual está diseñada para ejecutar la plataforma *.LRN* sobre el sistema operativo *Ubuntu 8.04*, y todo el software que incluye es *OpenSource*.

[8] https://gradient.it.uc3m.es/xowiki/dotlrn_vm

Esta máquina virtual está basada en el sistema de virtualización por software *VMware*[9], desarrollado por la empresa privada *VMware Inc.*. Para su ejecución es necesario utilizar el programa *VMware Player*, disponible de manera gratuita para los sistemas operativos Windows y Linux o bien el programa *VMware Fusion* disponible para ordenadores Macintosh con procesadores Intel. Este último aunque es similar al anterior, no es gratuito. La licencia de uso de la última versión disponible es de 79,99 dólares.

3.3.2. Git

Git es un sistema de control de versiones de código abierto distribuido mediante licencia *GNUv2*. Fue inicialmente desarrollado por Linus Torvalds para la realización del *kernel* de *Linux*. Se diseñó haciendo especial énfasis en la eficiencia y velocidad, sobre todo para proyectos grandes en los que se crean y unen un gran número de ramas, ya que guarda de manera eficiente cada uno de los estados por los que ha pasado, así como el autor de cada uno de los cambios .

La principal alternativa de *Git* es *Apache Subversion* (*SVN*), también *OpenSource*. Éste se diferencia principalmente en que, mientras *SVN* es centralizado, *Git* ofrece un modelo *P2P*. El mayor inconveniente de *Git* respecto a *SVN* es su mayor curva de aprendizaje, ya que *SVN* es más fácilmente utilizable [3] [2].

Para la realización de este proyecto, se ha elegido *Git* para mantener un control de versiones, pero también para disponer de una copia de seguridad. Para ello, se ha utilizado un servidor de la Universidad, en el que se han ido copiando las sucesivas versiones del proyecto utilizando el comando `push`.

Se hubiera podido utilizar igualmente *SVN* como controlador de versiones, ya que al tratarse de un proyecto de magnitud reducida y realizada por un único desarrollador, no tienen gran importancia las ventajas de *Git* sobre *SVN*. Sin embargo, se ha considerado más interesante aprender la utilización de *Git*.

3.3.3. Firebug

Firebug es una extensión del navegador *Firefox* que permite analizar la estructura del código *HTML* de una página *web*. Es *Open Source* y se distribuye de manera gratuita mediante licencia *BSD*. Al utilizarlo se puede ver claramente la estructura *DOM* del código, y observar el efecto

[9] http://www.vmware.com/

que tiene la modificación de los elementos sobre la forma en que se presenta la página.

Se ha empleado principalmente para analizar la estructura de los formularios proporcionados por *Google*, pero también para depurar las páginas generadas por la aplicación.

3.3.4. Wget

Wget es una herramienta gratuita y *Open Source*, distribuida mediante licencia *GPLv3*, que permite realizar peticiones a un servidor *web* utilizando protocolos *HTTP*, *HTTPS* y *FTP* con todo tipo de parámetros.

Wget se ha utilizado en el desarrollo de esta aplicación para comprobar el funcionamiento de las conexiones con el servidor de *Google Docs*. Esto ha sido necesario porque la librería *TCLcurl*, con la que se realizan las peticiones, era utilizada por primera vez por el programador. Al existir errores en el proceso de autenticación en el servidor de Google, fue necesario discernir si dichos errores eran debidos a una mala interpretación del *API* de *Google Docs*, o al de *TclCurl*. Al probar a realizar las peticiones mediante *Wget*, se ha podido comprobar que se había entendido correctamente el funcionamiento descrito en el *API* de *Google Docs*, y el problema era la utilización de *TclCurl*.

3.3.5. Wireshark

Wireshark es un analizador de tráfico gratuito, *Open Source* y multiplataforma, distribuido mediante licencia *GPL*, que captura todos los paquetes enviados y recibidos a través de la interfaz de red y muestra cada uno de sus parámetros para la mayoría de protocolos.

Junto con *Wget*, ha permitido comprobar la correcta comprensión del modo de comunicación con el servidor de *Google*. También ha sido utilizado para comprender el funcionamiento de algunas páginas como `http://spreadsheets.google.com/viewanalytics?formkey=claveForm`, en la que es necesaria una segunda petición para obtener los datos necesarios para mostrar dicha página. Dicha petición estaba especificada en el código *javascript*, y ha sido imprescindible visualizar el tráfico que realiza un navegador, para descubrir la necesidad de llevar a cabo una segunda petición *HTTP*, sin la cual la página sería mostrada sin contenido.

3.3.6. Selenium IDE

Selenium es una herramienta para realizar pruebas a aplicaciones *web*. *Selenium IDE* es un Entorno de Desarrollo Integrado para el desarrollo de dichas pruebas implementado como una extensión de *Firefox*. Es distribuido gratuitamente mediante la licencia *'Apache License'*, compatible con *GPLv3*. Permite almacenar una batería de acciones realizadas por el usuario, para luego volver a repetirlas sin que el usuario tenga que ejecutarlas una por una.

Capítulo 4

Descripción de la aplicación

En este capítulo se abordará toda la descripción de la aplicación, cual es su estructura y cómo se ha diseñado teniendo en cuenta las posibilidades y limitaciones de las tecnologías utilizadas que fueron descritas en el capítulo anterior.

4.1. Descripción general del sistema

Antes de proceder a describir la estructura de la aplicación, se explicará en está sección en qué consiste el sistema, de una manera más detallada que en el capítulo de introducción.

4.1.1. Finalidad de la aplicación

La finalidad de la aplicación desarrollada consiste en integrar el servicio de Formularios de *Google* dentro de la plataforma *.LRN*, añadiendo la funcionalidad de que el acceso a dichos formularios esté restringido a ciertos usuarios y sea posible identificar qué usuario realizó cada una de las respuestas del formulario. Ambas funciones no las ofrece el servicio de formularios de *Google*, el cual permite a cualquier usuario anónimo responder a los formularios un número indefinido de veces.

Los requisitos que debe de cumplir un usuario para poder realizar los formularios de la aplicación son los siguientes:

- Pertenecer a la comunidad de *.LRN* a la que se ha asignado dicho formulario.
- No haber realizado el formulario el número máximo de veces permitido para el mismo.

- Encontrarse dentro del plazo de realización de dicho formulario.

4.1.2. Modelo de usuario

Se distinguen dos tipos distintos de usuarios en la aplicación: administrador y usuario (no administrador).

- Administrador. Tiene acceso a todas las funcionalidades de la aplicación.

  Las funcionalidades consisten en gestionar, tanto los parámetros de los formularios (incluyendo añadir nuevos) como las respuestas recibidas a dicho formulario. En la sección 4.1.3 se explicarán en detalle en qué consisten dichas funcionalidades.

  En el entorno educativo de *.LRN* el administrador suele ser el profesor. A pesar de que en *.LRN* existen usuarios con un rol de profesor que pueden no ser administradores, se ha considerado que para hacer uso de las funciones de la aplicación debe de ser necesario tal permiso. Por tanto, todo aquel profesor que quiera hacer estas tareas deberá de adquirir los derechos de administración para esta aplicación.

  Además del administrador que maneja la aplicación, debe de existir un administrador que puede ser el mismo o distinto, encargado de instalar el módulo e instanciarlo en el curso.

- Usuario. Únicamente puede realizar formularios (si reúne los requisitos).

  Está pensado que sean aquellos usuarios que desempeñen el rol de alumnos, aunque al igual que en el caso anterior, no se consideran los roles asignados en la comunidad de *.LRN*, por lo que cualquier usuario que pertenezca a la comunidad puede realizar formularios con independencia de su rol.

4.1.3. Funcionalidades

Las funcionalidades que ofrece la aplicación a todos los usuarios son las siguientes:

- Mostrar una lista de formularios disponibles para la comunidad de usuarios a la que se accede. En dicha lista se muestran los siguientes datos del formulario: nombre, descripción, fecha de inicio de plazo, fecha límite y número de veces que se puede realizar. Un enlace en cada elemento de la lista permite acceder a la siguiente funcionalidad.

- Realizar aquellos formularios para los que se tenga permiso. Esta es la funcionalidad principal de la aplicación. Se encargará de que el usuario pueda realizar el formulario de manera que no pueda conocer la *URL* pública que permite realizar dicho formulario sin acceder a través de la aplicación, y autenticando al usuario como autor de dicha respuesta del cuestionario.

Los usuarios con permiso de administración tienen además acceso al resto de las características de la aplicación, las cuales se detallan a continuación:

- Ver la lista de formularios existentes en su cuenta de *Google Docs*, y poder, a partir de dicha lista, acceder a la opción de añadirlos a la comunidad sin necesidad de insertar manualmente su *URL*.

- Añadir un formulario de *Google* a la lista de formularios de la comunidad. El usuario puede acceder a esta opción directamente, en cuyo caso tendrá que proporcionar manualmente la *URL* del formulario y el texto de la pregunta que se vaya a usarse para identificar a los usuarios que respondan; o bien acceder a través de la lista de formularios.

- Editar los parámetros de un formulario, a excepción de su *URL* (que contiene su clave pública). Mediante esta opción se puede cambiar el nombre y descripción, modificar el plazo de resolución, el número de realizaciones disponibles y el texto de la pregunta del formulario utilizado para identificar a los usuarios. Esto último es de utilidad en el caso de que se edite el formulario de *Google* cambiando el nombre de dicha pregunta.

- Eliminar formularios añadidos a la comunidad, de manera irreversible. Tanto a esta característica como a la anterior se puede acceder desde la lista de formularios disponibles.

- Mostrar una lista de todas las respuestas de un formulario realizadas a través de la aplicación, identificando qué usuario realizó cada respuesta, para el caso de formularios autenticados. En esta misma lista existe la opción de mostrar las respuestas no realizadas a través de la aplicación, advirtiendo de que no están autenticadas.

- Descargar dicha lista de respuestas en un fichero en formato *CSV*.

- Eliminar de la *Spreadsheet* en la que se almacenan las respuestas del formulario, todas aquellas que no se hayan realizado a través de la aplicación.

- Acceder directamente a la lista de estadísticas de las respuestas de un formulario.
- Acceder directamente a la página de creación de un nuevo cuestionario en *Google Forms*.

Se ofrece la posibilidad de que los formularios no lleven a cabo la autenticación y que por tanto los formularios sean anónimos, con el correspondiente problema de seguridad que plantea ante la posibilidad de que un usuario averigüe la *URL* pública del formulario. En tal caso, las características de mostrar la lista de respuestas autenticadas, descargar dicha lista y eliminar de la *Spreadsheet* las no autenticadas, no estarán disponibles para dicho formulario.

4.2. Estructura de archivos

Los archivos de la aplicación desarrollada siguen la estructura propia de todos los módulos de la plataforma *OpenACS*. En la figura 4.1 se muestra esta estructura.

A continuación se detalla la función de cada uno de los directorios y ficheros.

- `catalog/`

  Cómo se explicará en detalle en la sección 4.7, esta carpeta contiene los archivos necesarios para mostrar el texto en distintos idiomas.

  - `gforms.en\_US.ISO-8859-1.xml` y `gforms.es\_ES.ISO-8859-1.xml`

    Cada archivo de internacionalización contiene los textos en cada idioma distinto en que está localizada la aplicación.

- `lib/`

  Esta carpeta contiene páginas que pueden ser accedidas desde éste y otros paquetes.

  - `gforms-list.adp`, `gforms-list.tcl` y `gforms-list-postgresql.xql`

    La página `gforms-list` muestra la lista de formularios existentes para la instancia del paquete en que nos encontremos. Desde ella, los usuarios pueden realizar los formularios disponibles, y si son administradores realizar todas las funciones disponibles.

- `tcl/`

  Esta página contiene las librerías disponibles para ser utilizadas por todas las páginas de este y otros paquetes.

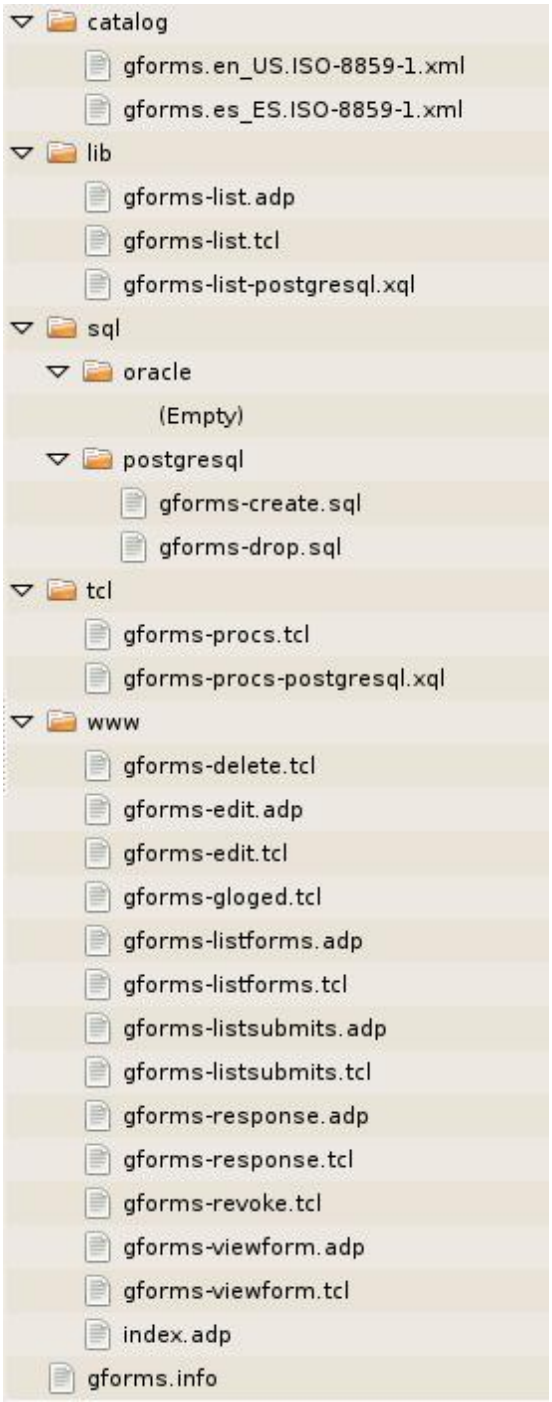


Figura 4.1: Estructura de archivos.

- `gforms-procs.tcl` y `gforms-procs-postgresql.xql`

  En este fichero *TCL* se han implementado los procedimientos utilizados para acceder a la base de datos, así como aquellos fragmentos de código que son reutilizados en más de una página.

- sql/

  Aquí se encuentran carpetas por cada uno de los lenguajes de base de datos. En dichas carpetas se guardan los *scripts* que deben ejecutarse al instalar y desinstalar la aplicación.

  - `oracle/`

    Cómo no se ha adaptado el paquete para su utilización en bases de datos *Oracle*, esta carpeta se encuentra vacía.

  - `postgresql/`

    Esta carpeta contiene los *scripts* para *PostgreSQL*

    - `gforms-create.sql`

      Contiene el código que debe ejecutarse al instalar el paquete, el cual crea las tablas y relaciones en la base de datos.

    - `gforms-drop.sql`

      Este *script* debe ejecutarse al desinstalar la aplicación, encargándose de eliminar de la base de datos todas las tablas, relaciones y objetos creados por el paquete que no vayan a ser utilizados por otros paquetes.

- `www/`

  Esta carpeta contiene todas las páginas que son visibles para los usuarios que tengan los permisos necesarios. Se puede acceder a estas páginas utilizando la *URL* en que se ha instanciado el paquete, seguido del nombre de la página. Si no se especifica el nombre de ninguna página se mostrará la página principal *index*.

  - `gforms-delete.tcl`

    Elimina el formulario especificado y a continuación redirecciona a la página principal.

  - `gforms-edit.adp` y `gforms-edit.tcl`

    Muestra una página en la que crear o editar un formulario, en función de si se recibe como parámetro su identificador.

- `gforms-gloged.tcl`

  Solicita a *Google* un *token* de sesión a partir del temporal que recibe como parámetro (`token`) y lo almacena en la base de datos. A continuación redirecciona a la página que haya recibido en otro parámetro (`go`).

- `gforms-listforms.adp` y `gforms-listforms.tcl`

  Muestra la lista de formularios que un usuario dispone en su cuenta de *Google Docs*, ofreciendo la posibilidad de añadir dichos formularios a la aplicación.

- `gforms-listsubmits.adp` y `gforms-listsubmits.tcl`

  Accede a la *Spreadsheet* correspondiente a un formulario y muestra los registros autenticados. Además, ofrece la funcionalidad de filtrar de la base de datos aquellos no autenticados y descargar un fichero *CSV* con las respuestas autenticadas del formulario.

- `gforms-response.adp` y `gforms-response.tcl`

  Realiza una petición de envío de respuesta de formulario a *Google*, con su correspondiente autenticación. Si la respuesta tenía campos obligatorios vacíos, vuelve a mostrar el formulario modificado cómo haría `gforms-viewform`.

- `gforms-revoke.tcl`

  Realiza una petición a *Google* para revocar el *token* existente y lo elimina de la base de datos. Después de dicha operación redirecciona a la página principal.

- `gforms-viewform.adp` y `gforms-viewform.tcl`

  Muestra un formulario modificado, si el usuario tiene permiso para verlo.

- `index.adp`

  Contiene la página principal, la cual se encuentra definida en `gforms-list`, por lo que redirecciona a dicha página (que se encuentra en la carpeta `lib`).

- `gforms.nfo`

  Este archivo almacena información sobre el paquete: su nombre en singular y plural, versión, datos del creador, los paquetes que requiere y una breve descripción sobre su funcionalidad. Si hubiera sido necesario, en este paquete se hubieran podido añadir parámetros para personalizarlo, así como procedimientos que fueran necesarios ejecutar durante su instalación y/o desinstalación (*callbacks*).

4.3. Utilización de la base de datos

En esta sección se explicará cómo se ha diseñado el modelo relacional de la base de datos. También se describirán cada una de las tablas que se han creado, así como los atributos que las componen y aquellos que se utilizan y que aparecen en otras tablas ya existentes en la plataforma *.LRN*. Por último, se explicará cómo se lleva a cabo el proceso de creación y eliminación de tablas y relaciones, así como el acceso y modificación de la base de datos.

4.3.1. Modelo relacional de la base de datos

El diseño del modelo relacional de la base de datos se ha realizado teniendo en cuenta el modelo de objetos existente en *OpenACS*, el cual amplía sus funcionalidades con el módulo *ACS Content Repository* (ya comentado en la sección 3.1.10), así como el modelo de usuarios integrado en dicho modelo de objetos.

El modelo de usuarios permite almacenar los datos de cada usuario, así como sus permisos. De esta manera, con una simple llamada a funciones ya definidas en la plataforma, podemos conocer el tipo de usuario que está accediendo al sistema y permitir el acceso a las funciones de administración únicamente a ciertos usuarios. Además, utilizando el modelo de objetos, no necesitamos almacenar en las tablas propias de la aplicación la información sobre los usuarios creadores de cada uno de los objetos, ni a qué paquetes pertenecen. Esto permite que la aplicación sea instanciada en varios paquetes, los cuales utilicen las mismas tablas propias de la aplicación, pudiéndose distinguir qué objeto pertenece a cada paquete, de manera que en cada instancia se manejen únicamente sus propios objetos.

El *Content Repository* nos aporta la posibilidad de guardar varias versiones (*revisions*) de un mismo ítem. Debido a que existe la funcionalidad de que un usuario administrador modifique las propiedades de un formulario, al realizar dicha acción se creará una nueva revisión en la base de datos, conservándose la anterior, lo cual permite conocer todos los cambios realizados.

Otra funcionalidad utilizada del *Content Repository* es la estructura jerárquica de los objetos, lo cual nos permite establecer relaciones jerárquicas (padre-hijos) entre los tipos de objetos que nosotros definamos. Esto es utilizado para la relación entre objetos de tipo formulario (`gforms`), que corresponderían con la función de padre, y los objetos que representan las respuestas de dichos formularios (`gfanswers`), que serían los hijos, al existir varios para un mismo padre.

En la figura 4.2 se muestra una representación de las tablas que se han diseñado para la

aplicación, teniendo en cuenta los atributos y relaciones necesarias en ellas, pero sin tener en cuenta el modelo real existente en la base de datos como consecuencia de la utilización del modelo de objetos del *Content Repository*.

Cómo se puede observar, los formularios necesitan su propia tabla en la que almacenar parámetros tales como número de intentos o fecha inicial y límite, de manera que se puedan cumplir los requisitos que se detallaron en la sección 4.1.1. Las respuestas de los formularios también necesitan su propia tabla, ya que es necesario almacenar qué *hash* se introduce en cada una, junto con el usuario que la realizó (ver sección 4.4.1). Por último, es necesario guardar los *tokens* de sesión utilizados por los usuarios administradores, para no solicitarles de nuevo el acceso en cada petición (ver sección 4.5.1).

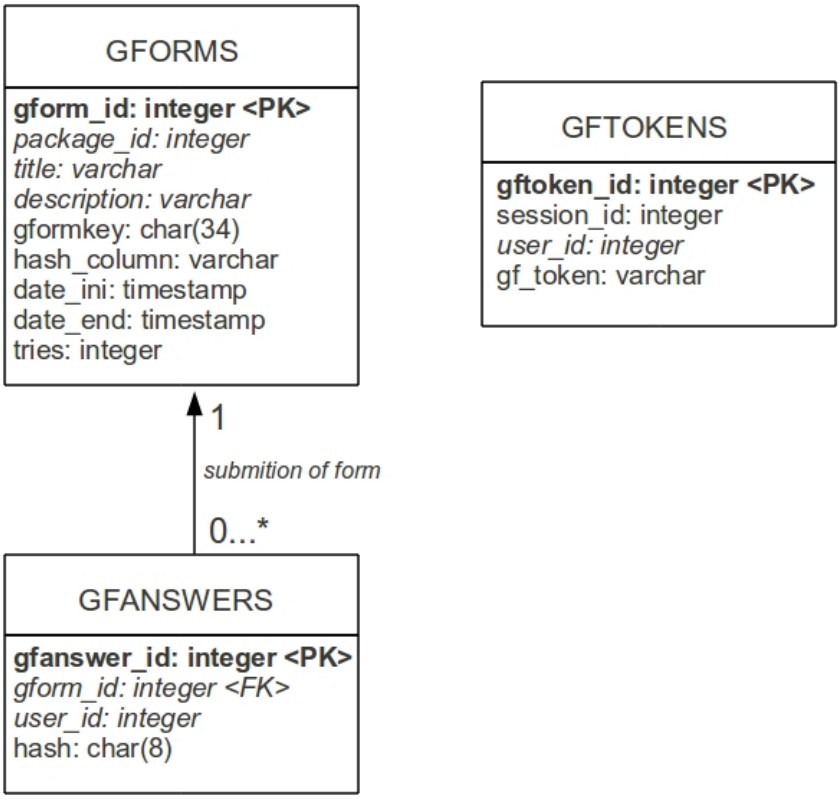


Figura 4.2: Diseño de modelo relacional de la base de datos (representación no real).

Los atributos dispuestos en cursiva son aquellos que no existen realmente en la tablas implementadas, al utilizarse los atributos correspondientes a dichos objetos ya existentes en las tablas `acs_objects`, `cr_items` y `cr_revisions`. De esta forma, aunque existe una relación entre

las tablas `gforms` y `gfanswers`, no existe dicha relación directa entre estas tablas, al no haber ningún atributo `gform_id` en la tabla `gfanswers` que referencia a `gforms`. Dicha *Foreign Key* se encuentra en la tabla `cr_items` del *Content Repository*, permitiendo que revisiones del tipo `gfanswers` referencien a padres del tipo `gforms`.

En la figura 4.3 se muestra el modelo relacional de la base de datos, incluyendo las tablas que se han definido para la aplicación junto con todas aquellas tablas y atributos, ya existentes en la plataforma *.LRN* que son utilizadas por la aplicación.

Cómo se puede observar, los atributos mencionados anteriormente en el diseño se encuentran en las tablas ofrecidas por el modelado de objetos del *Content Repository*. En la tabla 4.1 se muestra la correspondencia entre dichos atributos.

Tipo de objeto	**Atributo**	**Tabla**	**Nombre de atributo**
gforms	name	cr_revisions	title
gforms	description	cr_revisions	description
gforms	package_id	acs_objects	package_id
gfanswers	gform_id	cr_items	parent_id
gfanswers	user_id	acs_objects	creation_user
gftokens	user_id	acs_objects	creation_user

Tabla 4.1: Atributos utilizados en tablas de *OpenACS*.

4.3.2. Descripción de las tablas

Se han escogido los nombres de las tablas y atributos siguiendo el convenio existente: ambas deben ser descriptivas de su contenido, deben de estar en idioma inglés y los nombres de tablas deben de estar en plural. No obstante, para no dejar dudas acerca de la comprensión de lo que representa cada tabla y atributo, a continuación se dará una breve descripción.

Tabla gforms

Esta tabla contiene información sobre cada uno de los formularios añadidos a la aplicación. La tabla consta de las siguientes columnas, ninguna de las cuales puede ser nula.

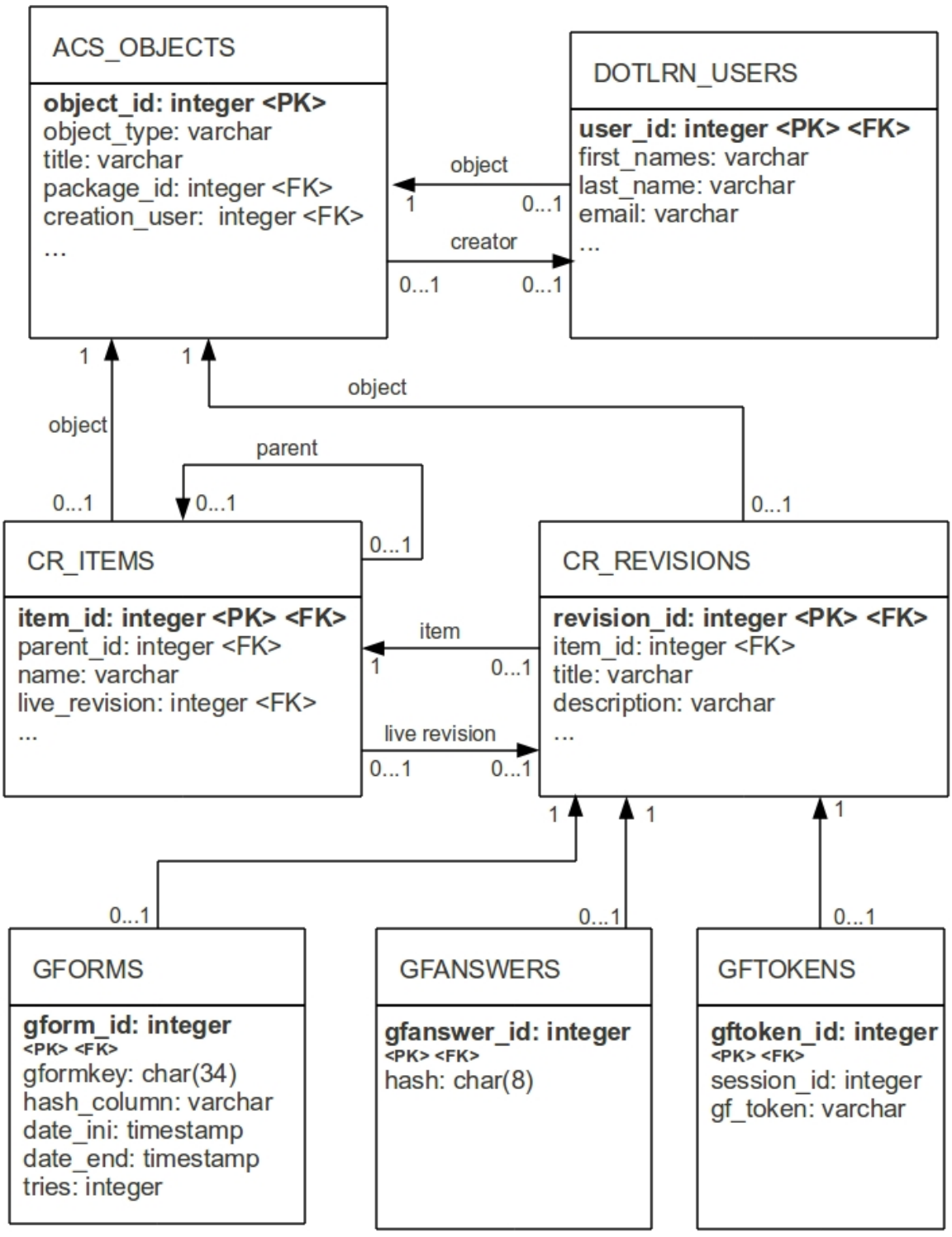


Figura 4.3: Modelo relacional de la base de datos.

- `gform_id`: integer, *primary key*.

  Es el identificador del objeto de la revisión del formulario. No debe de confundirse con el identificador del ítem del formulario, el cual representa a un formulario. Un formulario puede tener varias revisiones al haber sido editado, cada una con un distinto `gform_id`, pero compartiendo un mismo `item_id`.

- `gformkey`: char(34), único.

  Es la clave pública del formulario (`formkey`). Se ha decidido no almacenar la clave de su *Spreadsheet* asociada, siendo necesario buscarla entre la lista de *Spreadsheets* del usuario cada vez que se quiera acceder a ella, con la finalidad de que puedan ser añadidos formularios por parte de administradores que no tengan permisos para acceder a dicha *Spreadsheet* en su cuenta de *Google Docs*.

- `hash_column`: varchar.

  Almacena el nombre de la pregunta del formulario en la cual se almacenará el *hash* identificador de usuario. Se ha preferido almacenar el nombre de la pregunta en lugar del valor del parámetro `name` utilizado por el formulario (ver sección 4.4.3), ya que en caso de editar el formulario es posible que el valor del parámetro cambie, y de esta forma queda garantizado el funcionamiento después de modificarse, siempre y cuando el usuario mantenga el nombre de la pregunta.

- `date_ini`: timestamp.

  Fecha y hora a partir de la cual es posible realizar el cuestionario. Hasta ese momento el cuestionario no estará accesible.

- `date_end`: timestamp.

  Fecha y hora límite para realizar el cuestionario. A partir de dicho momento no será posible realizar el cuestionario.

- `tries`: integer.

  Número de veces que cada usuario puede realizar el cuestionario.

Además de estas columnas, por el hecho de tener asociado cada registro de esta tabla un registro correspondiente de las tablas `cr_revisions` y `acs_objects`, también son utilizados los

siguientes atributos que aparecen en las columnas de las tablas citadas para los elementos de esta tabla.

- `package_id`: integer.

  Identifica al paquete en que se encuentra instanciada la aplicación. De esta forma, es posible crear varias instancias del paquete, cada una con sus propios formularios. En cada instancia de la aplicación aparecerán únicamente los formularios de dicha instancia, a pesar de estar todos almacenados en la misma tabla.

- `title`: varchar.

  Es el nombre del formulario que se mostrará públicamente a todos los usuarios.

- `description`: varchar.

  Además del nombre, los usuarios pueden identificar al formulario con esta descripción.

Tabla gfanswers

Es la tabla que almacena un registro cada uno de las veces que alguien envía un cuestionario ('respuesta'). Es la única tabla en la que pueden añadir registros los usuarios que no son administradores, al realizar formularios. Se compone de las siguientes columnas.

- `gfanswer_id`: integer, *primary key*.

  Al igual que en el caso anterior, se corresponde con la revisión del elemento. A pesar de no existir la opción de crear más revisiones de un envío de cuestionario, por el hecho de utilizar el *Content Repository* se ha decidido seguir el mismo proceso.

- `hash`: char(8).

  Almacena el *hash* que ha sido enviado en la respuesta del cuestionario para posteriormente poder identificar el registro en la *Spreadsheet*.

Al igual que en el caso anterior, debido a que tiene una entrada correspondiente en las tablas `cr_items` y `acs_objects`, son usados los siguientes atributos almacenados en columnas de dichas tablas.

- `parent_id`: integer.

  Es la *foreign key* que identifica al formulario al que pertenece la respuesta al cuestionario.

- `creation_user`: integer.

  Identifica al usuario que creó el objeto, y que por tanto realizó la 'respuesta' del cuestionario. Constituye una *foreign key* hacia las tablas de usuarios `users` y `dotlrn_users`, siendo utilizada esta última para conocer los datos de dicho usuario (nombre, email, etcétera).

Tabla gftokens

Esta tabla se decidió añadir en las últimas versiones de la aplicación, debido a la necesidad de almacenar los *tokens* de sesión para la utilización del *API* de *Google Docs*. En las primeras versiones, antes de utilizar esta tabla, se solicitaba al usuario un nuevo *token* cada vez que quería acceder a una función que lo necesitara. El *token* de sesión era pasado como un parámetro de *URL*, de la misma manera que se recibe de *Google* a lo largo de sucesivas páginas, pero cada vez que se volvía a la lista de formularios y se volvía acceder a una funcionalidad que lo requiriera, era necesario solicitar un nuevo *token*, ya que este no quedaba almacenado en ninguna parte.

Para solucionar este problema se ha añadido esta tabla, de forma que el *token* quede almacenado en la base de datos junto con el identificador de la sesión del usuario en *.LRN* en la cual se utilizó. La finalidad de almacenar la sesión en la que se utiliza es volver a solicitar el permiso al usuario cada vez que inicie una nueva sesión, ya que no se ha considerado 'amigable' reutilizar el permiso que se concedió en otra sesión. No obstante, esto es una decisión de diseño discutible, ya que algunas aplicaciones se diseñan para que se utilicen los permisos concedidos por el *token* mientras éste sea válido (los *tokens* de *Authsub* no caducan pero sí expiran si se revocan manualmente o una aplicación solicita más de 10 para el mismo usuario).

Las columnas de esta tabla son las siguientes.

- `gftoken_id`: integer, *primary key*.

  Al igual que en los casos anteriores esta clave primaria se corresponde con el elemento de la tabla `cr_revisions`.

- `session_id`: integer.

  Identifica la sesión en la que se utiliza el token.

- `token`: varchar.

  Es el *token* de sesión de *Google Docs*, utilizado para realizar peticiones sobre las *Spreadsheets* del usuario, tales como obtener todos los registros de la que se corresponde con un

determinado formulario.

El identificador de usuario es almacenado, al igual que en el caso anterior, en su correspondiente registro de la tabla `acs_objects`.

- `creation_user`: integer.

  Identifica al usuario propietario de dicho *token*.

Para evitar que la tabla pueda crecer indefinidamente y al no tener ninguna utilidad utilizar los *tokens* antiguos, a pesar de utilizar el *Content Repository* por las facilidades del manejo de objetos que aporta, cada vez que se introduce un nuevo registro en la tabla de un usuario que ya tenía otro registro, el registro más antiguo es borrado (su revisión y su ítem).

4.3.3. Creación de tablas

Para la creación de tablas, relaciones entre éstas (proceso necesario al instalar la aplicación), así como su correspondiente eliminación (proceso necesario al desinstalar la aplicación), se han definido los ficheros `gforms-create.sql` y `gforms-drop.sql`.

En estos ficheros en lugar de utilizar llamadas directas a la base de datos, en su correspondiente lenguaje, se han utilizado funciones proporcionadas por el *Content Repository* para llevar a cabo la creación y eliminación de tablas, así como la creación de los tipos correspondientes a dichas tablas y las relaciones jerárquicas existentes entre ellas (los objetos de `gfanswers` tienen su correspondiente padre de tipo `gforms`).

Aunque el modelo relacional de la base de datos sea complejo por la utilización del *Content Repository*, éste se encarga de crear unas vistas con el mismo nombre de cada nueva tabla añadiendo una `x` (`gformsx`, `gfanswersx` y `gftokensx`). Estas vistas serán utilizadas para acceder a los elementos actuales (*live*) de cada tabla, de manera que tengamos en dichas vistas las columnas que se ha mencionado anteriormente (`title`, `description`, `package_id`, `parent_id` y `creation_user`) que se encuentran en las tablas `acs_objects`, `cr_items` y `acs_objects`.

4.3.4. Acceso a la base de datos

Cómo se ha comentado anteriormente, *OpenACS* soporta bases de datos *PostgreSQL* y *Oracle*, habiéndose desarrollado la aplicación para bases de datos *PostgreSQL* únicamente. Esto limita el uso de la aplicación a aquellos servidores que utilicen bases de datos *PostgreSQL*, por ello, para

permitir una fácil adaptación de la aplicación a entornos que utilicen otra base de datos, todas las llamadas a la base de datos (*queries*) se han almacenado en ficheros independientes. De esta manera, para utilizar otra base de datos tan sólo sería necesario crear los archivos que contienen las *queries* en el lenguaje que utilice la otra base de datos.

Se han creado en el fichero `gforms-procs.tcl` todos los procedimientos de acceso y manipulación utilizados por todas las páginas de la aplicación a excepción de `gforms-list.tcl`, que accede directamente a la base de datos. Existen por tanto dos ficheros en los que se almacenan las *queries* de la base de datos, uno para cada estos dos archivos.

Debido al complejo tratamiento de los objetos necesario al utilizar el *Content Repository*, el cual utiliza cuatro tablas para almacenar cada elemento (`acs_objects`, `cr_items`, `cr_revisions` y la propia de cada objeto), a continuación se explica el proceso necesario para crear, editar y eliminar objetos de la base de datos. El resto de operaciones con la base de datos se consideran relativamente sencillas si se conoce el modelo relacional, por lo que no se considera relevante su explicación.

Añadir un elemento a la base de datos

El proceso para añadir un elemento de tipo `gforms`, `gfanswers` o `gftokens` es el siguiente:

- Crear un nuevo ítem (tabla `cr_items` y su correspondiente objeto en `acs_objects`). Se utiliza la función correspondiente proporcionada por el *Content Repository*, la cual requiere asignar un nombre único al ítem. Además, para el caso de `gf_answers` se asigna como `parent_id` la clave que identifica al ítem de su formulario correspondiente.

- Crear una nueva revisión (tabla `cr_revisions` y su correspondiente en `acs_objects`). De igual forma se utiliza la función correspondiente del *Content Repository*. Para el caso de *gforms* se crea la revisión con el título (`title`)y la descripción (`description`) del formulario.

- Añadir la entrada en la tabla propia de la aplicación (`gforms`, `gfanswers` o `gftokens`). En este caso es en el que es necesario el acceso a la base de datos utilizando las *queries* que se han definido en los ficheros independientes.

- Asignar como revisión actual la que hemos creado. Para ello se utilizará la función correspondiente del *Content Repository*.

Modificar un elemento a la base de datos

La modificación de elementos solamente se realiza para aquellos de la tabla `gforms`, en el caso en que el administrador quiera editar las propiedades de un formulario. El procedimiento es igual que en el caso de creación de un nuevo elemento sólo que en este caso no se crea un nuevo ítem (lo cual era el primer paso).

Eliminar un elemento de la base de datos

En caso de que queramos eliminar un formulario (`gforms`) o *token* (`gftokens`) el proceso a seguir es el siguiente.

- Asignar al ítem (`cr_items`) como revisión actual (*live*) ninguna. Se utiliza la función correspondiente proporcionada por el *Content Repository*,
- Eliminar el registro (o registros) existentes en la tabla (`gforms` o `gftokens`) correspondiente con dicho ítem.
- Eliminar la revisión (o revisiones) (tabla `cr_revisions` y su correspondiente `acs_objects`).
- Eliminar el ítem (tabla `cr_items` y su correspondiente `acs_objects`).

4.4. Proceso de realización de cuestionarios

En la sección 3.2.1 del capítulo anterior se explicó en detalle el funcionamiento de la aplicación de Formularios de *Google Docs*, consistente en que la página *spreadsheets.google.com/viewform* muestra un formulario, el cual envía los datos a la página *spreadsheets.google.com/formResponse*, y ésta añade el registro a la *Spreadsheet* si todos los campos obligatorios fueron rellenados, o vuelve a mostrar el formulario si no se recibieron todos los campos requeridos.

La idea inicial de implementación de la aplicación consistía en mostrar al usuario el formulario proporcionado por *spreadsheets.google.com/viewform* añadiendo únicamente un campo invisible con un código *hash* que identificara al usuario. Esto suponía una limitación importante, ya que en el caso de los formularios con campos obligatorios, si hay campos sin rellenar, la página *spreadsheets.google.com/formResponse* muestra de nuevo el formulario, pero al ser obtenido directamente de *google.com*, el formulario no contiene el campo oculto con la identificación.

Debido a la importancia de tener compatibilidad con formularios que contienen campos obligatorios, se ha realizado un nuevo diseño de la implementación de la aplicación, el cual modifica no sólo los formularios que proporciona *spreadsheets.google.com/viewform*, sino también los proporcionados por *spreadsheets.google.com/formResponse*. En el caso de *formResponse* si se han introducido todos los campos obligatorios, la página no muestra ningún formulario, lo cual significa que fueron añadidos con éxito a la *Spreadsheet*. En dicho caso, deberemos registrar en la base de datos de la aplicación, qué usuario ha realizado un cuestionario y qué *hash* fue utilizado, para posteriormente poder identificarle.

La aplicación se ha diseñado de manera que existan dos páginas que acceden respectivamente a *viewform* y *formResponse*, y muestran los formularios modificados. Estas páginas se han denominado en la aplicación de manera similar al nombre que tienen en *Google*: `gforms-viewform` para el caso de *viewform* y `gforms-response` para el caso de *formResponse*.

La página `gforms-viewform` se encarga en primer lugar de comprobar los permisos del usuario para la realización del formulario: si pertenece a la comunidad, si no ha resuelto dicho formulario el número máximo de veces y si el momento actual se encuentra dentro del plazo para la resolución del cuestionario. A continuación solicita a *Google* el formulario, lo modifica tal y cómo se explicará en la sección 4.4.2 y lo muestra al usuario.

La página `gforms-response` resulta algo más compleja, ya que se encarga de introducir los datos de la respuesta del formularios en *Google*, y en función de la respuesta recibida mostrar otra vez el formulario modificado o añadir la entrada en la base de datos. Su funcionamiento se explicará en detalle en la sección 4.4.3.

De esta manera, el usuario nunca accede directamente al servidor de *Google* para enviar los formularios. En lugar de emplear la clave pública de Google para identificar al formulario, en nuestra aplicación utilizaremos el identificador del ítem del formulario (`item_id`). Esto aporta una ventaja importante, ya que al no proporcionar en ningún momento al usuario la clave pública del formulario, éste no podrá realizar el formulario sin utilizar la aplicación de *.LRN* (siempre que no descubra la clave pública utilizando otros medios). Esto le impide realizar posibles ataques, tales como volver a realizar un envío con el mismo *hash* (en el caso de que se le hubiera proporcionado al usuario en el código fuente de la página como se diseño en un principio), o realizar un envío masivo utilizando *hashes* aleatorios.

Una desventaja de esta implementación es la posible violación de las condiciones de uso, ya que el acceso en este caso lo realiza el servidor de la plataforma *.LRN*, y no a través del navegador

del usuario. Esto se explicará en detalle en la sección 4.8.

En la figura 4.4 se muestra un esquema de las peticiones *HTTP* que son realizadas por cada una de las tres entidades (usuario, servidor *.LRN* y servidor *Google*), para un ejemplo en el que el formulario tiene campos obligatorios, y el usuario olvida rellenar uno de estos campos la primera vez, completándolos la segunda vez que se le muestra el formulario.

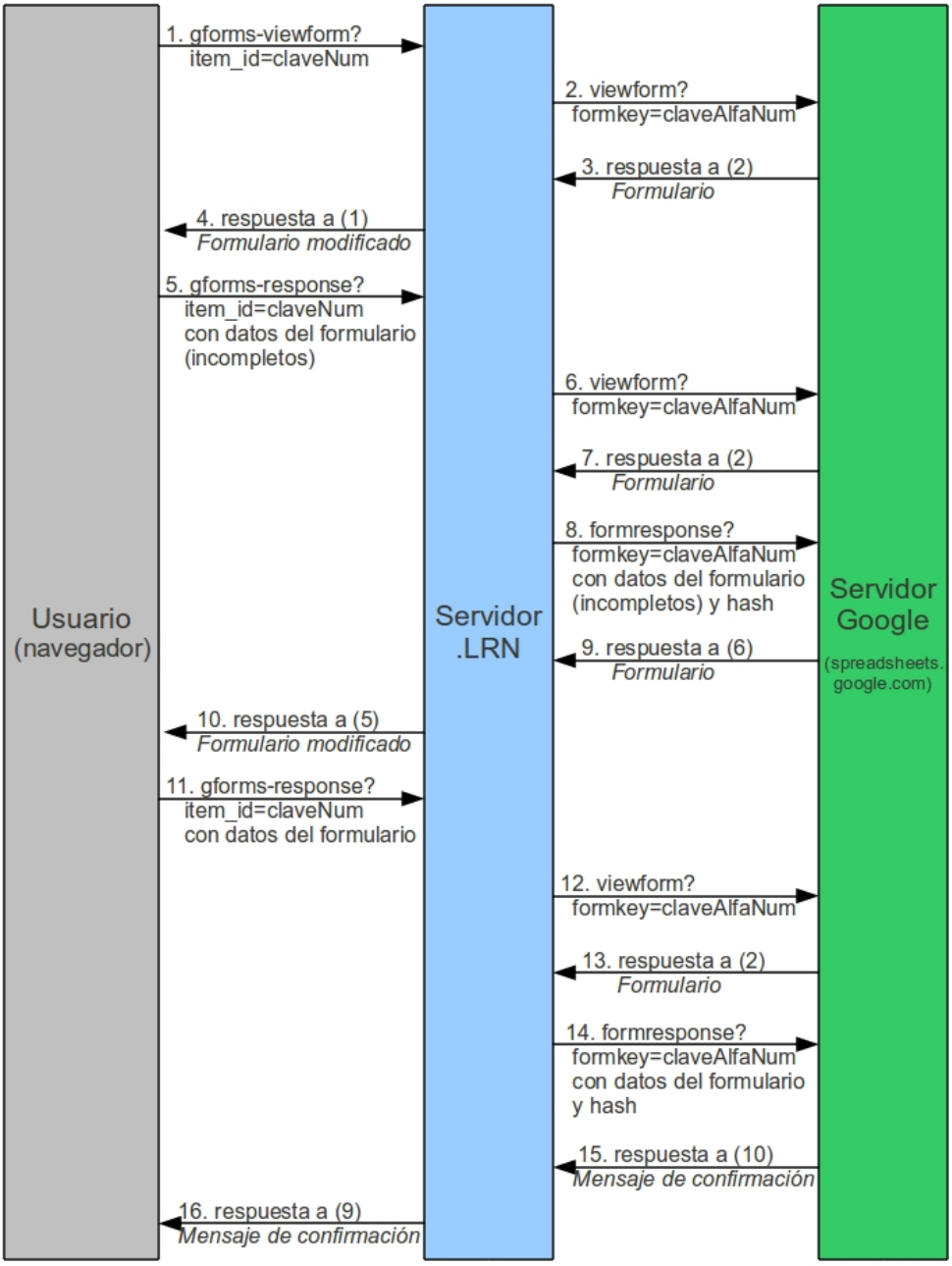


Figura 4.4: Esquema de peticiones en la realización de un formulario.

Cómo se puede observar, el usuario en ningún momento realiza ninguna petición al servidor de *Google*, por lo que es posible no proporcionarle en ningún momento la clave de acceso al formulario.

4.4.1. Identificación y autenticación de las respuestas

Para llevar a cabo una identificación del usuario que realizó cada una de las respuestas de los cuestionarios que aparecen en la *Spreadsheet* de *Google* se ha utilizado uno de los campos de las preguntas que componen el formulario de *Google*. Dicha pregunta será eliminada del formulario que se muestra al usuario, cómo se explicará en la siguiente sección 4.4.2.

Aunque la aplicación no le ofrece al usuario en ningún momento la posibilidad de conocer la clave (`formkey`) con la que acceder a dicho formulario desde *Google*, el acceso al formulario es público, y el usuario podría descubrir dicha clave utilizando un motor de búsqueda con las palabras que figuren en el formulario, o de una manera más compleja realizando peticiones masivas a *Google* para descubrir los formularios existentes.

Por ello, se ha diseñado el modelo de autenticación de las respuestas teniendo en cuenta la posibilidad de que un usuario pueda tener acceso al formulario público y en consecuencia realizar posibles ataques que suplanten identidad o envíen nuevas respuestas para un usuario que ya realizó el formulario.

El procedimiento que se ha llevado a cabo es la generación de un *hash* aleatorio para cada respuesta de un formulario. Dicho *hash* es añadido en el campo de la pregunta utilizada para la autenticación y, si el formulario es respondido con éxito (por haber rellenado todos los campos obligatorios), es almacenado en la tabla `gfanswers`, permitiendo identificar la respuesta con el usuario que la realizó.

Por comodidad a la hora de que el usuario utilice la *Spreadsheet*, en la columna del *hash*, junto con éste (en la misma celda) se añade el nombre del usuario que realizó la respuesta. Dicho nombre no resultará fiable en caso de que se haya hecho pública la clave del formulario, a menos que se realice el proceso de filtrado (ver sección 4.5.3)

El *hash* se encuentra formado por caracteres únicamente alfanuméricos sencillos (ni `ñ` ni acentos), para poder ser enviados a través de la petición *HTTP* sin necesidad de codificarlos en formato *URL* (lo cual es necesario para caracteres especiales). Su longitud se ha diseñado teniendo en cuenta la posibilidad de que se realice un ataque masivo que envíe respuestas con

hashes aleatorios, tal y cómo se explica a continuación.

De acuerdo con las especificaciones de las *Spreadsheets* de *Google*[1], no existe un número máximo de filas en una *Spreadsheet*, pero sí un máximo de 400.000 celdas. El peor de los casos sería aquél en el que un formulario que tuviera una única pregunta, la cual sería utilizada para la autenticación, por lo que el usuario no podría introducir ningún dato en el formulario que no fuera el hecho de enviarlo, pero que tendría su utilidad en el caso en el que se pidiera a los usuarios únicamente inscribirse, apareciendo su registro en la *Spreadsheet*. Además de la pregunta, *Google Formularios* añade una columna denominada 'Marca temporal' para almacenar la fecha y hora de cada respuesta, por lo que deben de existir como mínimo dos columnas en la *Spreadsheet* de un formulario. Esto significaría un máximo de 200.000 filas en la *Spreadsheet*, que sería el número máximo de veces que podría ser respondido un formulario.

Para calcular el número de caracteres que debe tener el *hash* que garantice que ante la existencia de un ataque no se genere ningún *hash* que coincida con los ya existentes, en un principio se pensó en calcular la probabilidad que existiría de que ocurriera al menos una coincidencia, para *hash* de un número determinado de caracteres. En dicho caso, la probabilidad depende además del número de filas válidas y las que quedan disponibles para realizar el ataque. Al realizar dichos cálculos se ha observado que el peor de los casos se daba cuando la mitad de las filas ya contenían registros y la otra mitad quedaban disponibles para el ataque. No se entrará en detalle en estos cálculos, ya que no es el procedimiento que se ha llevado a cabo.

En una versión posterior se pensó en la posibilidad de que, de igual forma que los supuestos *hashes* creados ante un posible ataque no tuvieran suficiente probabilidad de coincidir con los ya existentes, los creados por la aplicación tampoco tendrían dicha probabilidad de coincidir entre sí. De esta manera no es necesario llevar a cabo una consulta a la base de datos para verificar que el *hash* creado aleatoriamente no coincida con los ya existentes de dicho formulario. Así pues, lo que hay que considerar es que de las 200.000 posibles respuestas que admite un formulario, la probabilidad de que coincidan dos *hashes* sea mínima.

En criptografía se conoce como 'Ataque de cumpleaños' (*'Birthday attack'*) al posible ataque cómo consecuencia de que dos *hashes* de un conjunto coincidan. Para evitar este tipo de ataques se debe de tener en cuenta la probabilidad de que dos o más *hashes* coincidan. El cálculo exacto de dicha probabilidad es complejo debido a la magnitud de los números, por lo que se llevan a cabo

[1] http://docs.google.com/support/bin/answer.py?hl=en&answer=37603

aproximaciones [27] [13]. La expresión de la probabilidad de que existan dos o más *hashes* iguales en un conjunto de *'n'* elementos cuyos *hashes* pueden tomar *'d'* posibles valores equiprobables se muestra en la figura 4.5.

$$P(n,d) = 1 - \frac{d!}{(d-n)!d^n} \approx 1 - e^{-n(n-1)/2d} \approx 1 - \left(1 - \frac{n}{2d}\right)^{n-1} \tag{4.1}$$

Figura 4.5: Probabilidad de que coincidan a menos dos *hashes*.

En nuestro caso, *'n'* tomaría el valor del número de filas posibles (200.000), y *'d'* el número de hashes posibles, que sería el número de valores distintos que puede tomar cada carácter (62) elevado al número de caracteres que componen el *hash*.

En la tabla 4.2 se muestra la probabilidad de que existiera al menos una colisión entre los *hashes* en función del número de caracteres que lo formen, calculada utilizando la aproximación anterior.

Número de caracteres	**Número de *hashes* posibles**	**Probabilidad de coincidencia**
< 6	$< 10^9$	1
6	$5,68 \times 10^{10}$	$2,97 \times 10^{-1}$
7	$3,52 \times 10^{12}$	$5,66 \times 10^{-3}$
8	$2,18 \times 10^{14}$	$9,16 \times 10^{-5}$
9	$1,35 \times 10^{16}$	$1,48 \times 10^{-6}$
10	$8,39 \times 10^{17}$	$2,38 \times 10^{-8}$
11	$5,20 \times 10^{19}$	$3,77 \times 10^{-10}$
12	$3,12 \times 10^{21}$	$< 10^{-10}$

Tabla 4.2: Probabilidad de coincidencia entre dos o más *hashes*.

Se ha elegido que cada *hash* esté compuesto por 8 caracteres, pues con esta cantidad la probabilidad ya es inferior a 0,01 %, valor que se ha considerado razonable.

4.4.2. Modificación de formularios

La modificación de formularios, consiste principalmente en eliminar del formulario proporcionado por *Google* la pregunta correspondiente al campo en el que se va a introducir la autenticación del usuario y modificar la dirección de envío, de manera que en lugar de enviarse los datos a *Google*, se envíen a la página `gforms-response` de la aplicación.

Se ha creado un proceso denominado `gforms::mod_form` que realiza la función de modificar el formulario, de manera que dicho código sea reutilizable tanto por la página `gforms-viewform` como por la página `gforms-response`.

Para una correcta visualización del formulario, es necesario no solamente su fragmento correspondiente de código, sino aquél que contiene las hojas de estilo (*CSS*) ya que sin ellas no se distinguiría el texto de las preguntas del texto de ayuda, y la visualización no sería muy amigable. Por ese motivo, además del código del formulario se reutiliza el código de la cabecera. También se añade el pié de página original, eliminando en este caso un enlace para reportar un abuso del formulario, ya que contiene la clave pública del formulario que no debe de conocer el usuario.

En el caso de utilizar la funcionalidad de formularios anónimos, estos no serán modificados. Se mostrarán todas las preguntas al no existir ninguna destinada a la identificación del usuario, y no se podrá añadir el *hash* en ninguna columna, por lo que el formulario no será seguro ante la posibilidad de que un usuario descubra la clave pública del formulario (`formkey`).

A continuación se muestra como ejemplo un formulario sencillo que dispone de dos campos de texto, uno de una única línea (tipo *text*) y otro de varias lineas (tipo *textarea*). El de una única línea se corresponde con una pregunta a cuyo campo de texto se le ha asignado 'Hash' y será utilizado por la aplicación para llevar a cabo la identificación.

En la figura 4.6 se puede observar cómo es el formulario proporcionado por *Google*, el cual envía los datos a la página *formResponse* de su servidor e incluye las dos preguntas.

Cómo se puede observar en la figura 4.7, la aplicación elimina del formulario la pregunta correspondiente a la identificación y modifica la dirección de envío. Además, se añade un campo con el identificador del formulario utilizado por la aplicación, para que la página `gforms-response` pueda conocer el formulario que se está respondiendo.

```
<form id="ss-form" method="POST"
 action="https://spreadsheets.google.com/formResponse?formkey=...6MQ">
  ...
  <div ...>
    ...
      <label for="entry_0" class="ss-q-title">Hash</label>
      <label for="entry_0" class="ss-q-help"></label>
      <input type="text" id="entry_0" class="ss-q-short"
       value="" name="entry.0.single">
    ...
  </div>
  ...
  <div ...>
    ...
      <label for="entry_1" class="ss-q-title">Explica algo
        <span class="ss-required-asterisk">*</span>
      </label>
      <label for="entry_1" class="ss-q-help">en detalle</label>
      <textarea id="entry_1" class="ss-q-long" cols="75"
       rows="8" name="entry.1.single"></textarea>
    ...
  </div>
  <br>
  <input type="hidden" value="0" name="pageNumber">
  <input type="hidden" value="" name="backupCache">
  <div ...>
    ...
      <input type="submit" value="Enviar" name="submit">
    ...
  </div>
</form>
```

Figura 4.6: Formulario proporcionado por *Google*.

```
<form id="ss-form" method="POST" action="gforms-response">
  ...
  <div ...>
    ...
      <label for="entry_1" class="ss-q-title">Explica algo
        <span class="ss-required-asterisk">*</span>
      </label>
      <label for="entry_1" class="ss-q-help">en detalle</label>
      <textarea id="entry_1" class="ss-q-long" cols="75"
       rows="8" name="entry.1.single"></textarea>
    ...
  </div>
  <br>
  <input type="hidden" value="0" name="pageNumber">
  <input type="hidden" value="" name="backupCache">
  <div ...>
    ...
      <input type="submit" value="Enviar" name="submit">
    ...
  </div>
  <input type="hidden" value="9284" name="item_id">
</form>
```

Figura 4.7: Formulario modificado por la aplicación.

4.4.3. Envío de respuestas a *Google*

La página `gforms-response` se encarga de enviar los datos de la respuesta del formularios a *Google*, y en función de la respuesta recibida mostrar otra vez el formulario modificado o añadir la entrada en la base de datos. Para ello, cómo se muestra en la figura 4.4, en primer lugar obtiene la página del formulario *viewform*, y a partir de dicha página extrae el nombre del parámetro (`name`) cuya pregunta es utilizada para la identificación del usuario (en el ejemplo de la figura 4.6

sería `entry.0.single`). Al parámetro con dicho `name` le asignará el *hash* aleatorio, y se enviará junto con el resto de los parámetros recibidos a la página *formResponse*. Conviene destacar que el parámetro del *hash* aleatorio se sitúa en primer lugar en la lista de parámetros, lo cual permite ante un ataque del usuario que añada dicho parámetro, que el parámetro introducido por el usuario sea ignorado a la hora de añadirlo a la *Spreadsheet*.

Si la respuesta a la petición de *formResponse* contiene un formulario, significará que no se introdujeron todos los campos obligatorios y en consecuencia la respuesta no ha quedado registrada en la *Spreadsheet*. En dicho caso, se volverá a mostrar el formulario modificado, de igual manera que realiza `gforms-viewform`.

Si por el contrario, dicha petición contiene en lugar de un formulario, un mensaje de confirmación del registro en la *Spreadsheet* de las respuestas, la página `gforms-response` debe encargarse de registrar en la base de datos de la aplicación qué usuario ha realizado un cuestionario y qué *hash* fue utilizado, para posteriormente poder identificarle.

En la figura 4.8 se muestra un diagrama de flujo que aclara el funcionamiento de la página `gforms-response`.

4.5. Acceso a los datos de *Google Spreadsheets*

En esta sección se explica cómo obtener y modificar información contenido en la cuenta de *Google Spreadsheets* del usuario administrador. La mayoría de las funciones se realizarán utilizando el *API* de *Google Docs*/*Google Spreadsheets*, sin embargo para el caso de los formularios será necesario realizar además otra serie de peticiones, debido a la inexistencia de *API* de *Google Forms*.

4.5.1. Autenticación en el servidor de Google Docs

Cómo se explicó en la sección 3.2.3, se ha utilizado *AuthSub* como protocolo de autenticación en el servidor de *Google* para acceder a los datos de las *Spreadsheets* del usuario. La autorización mediante este protocolo requiere una secuencia en la cual interaccionan tres entidades: la aplicación *web* (servidor en que esté instalado *.LRN*), *Google* y el usuario. En la figura 4.9 se muestra el diagrama de dicha secuencia.

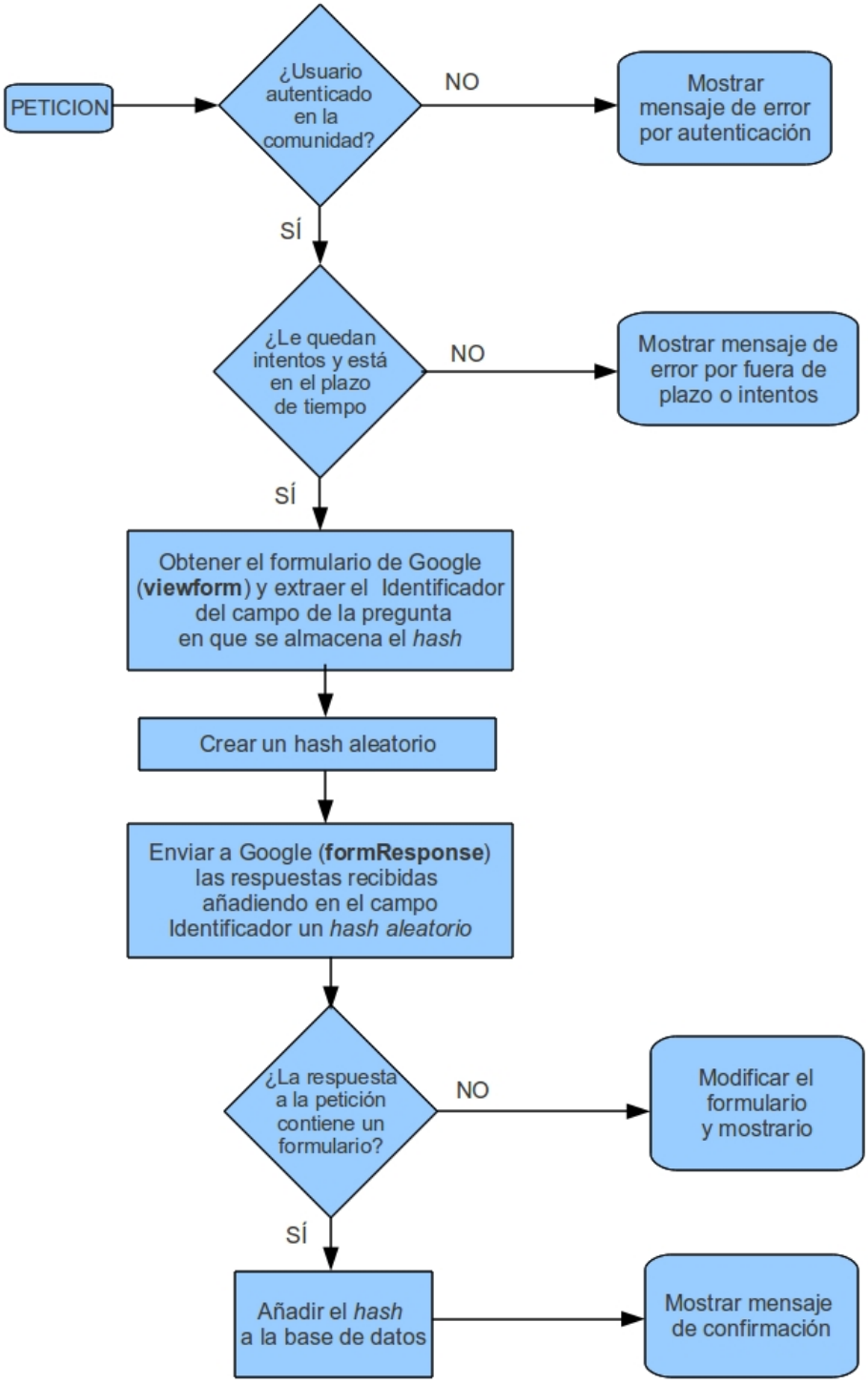


Figura 4.8: Diagrama de flujo de la página gforms-response.

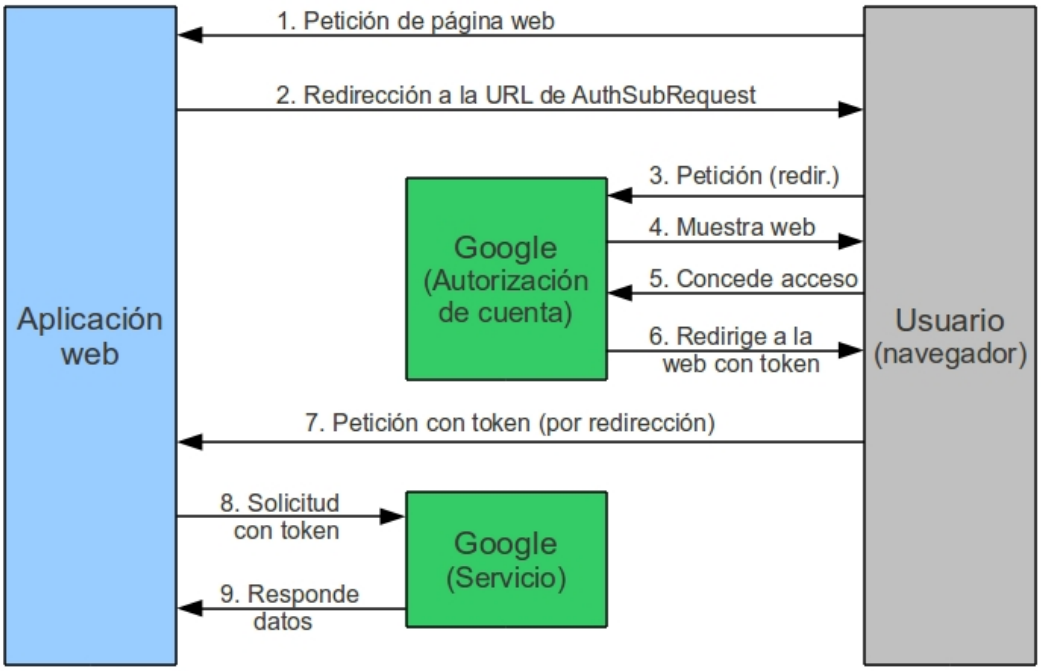


Figura 4.9: Esquema de peticiones para la obtención de token.

Cuando el usuario solicite acceder a una página que requiera la utilización de los *feeds* de *Google Spreadsheet* (*1*), y no se disponga de un *token* para dicho usuario, se le redireccionará a la página de autorización de *AuthSub* con unos parámetros `next`, `scope` y `session` determinados, los cuales se detallarán más adelante (*2*).

Será el navegador del usuario el que solicite a *Google* la página de *AuthSub* con los parámetros que determinó la aplicación (*3*). Esto se realizará de manera invisible para el usuario, al cual le aparecerá directamente la página de autorización (*4*).

En *5*, el usuario permite el acceso por parte de la aplicación a los datos de su cuenta de *Google* de un ámbito determinado el parámetro `scope` con que se accedió a dicha página, en este caso, *Google Spreadsheets*. Así pues, con dicho permiso, la aplicación no podría acceder a ningún otro dato de su cuenta fuera de dicho ámbito. Para permitir dicho acceso, el usuario tiene que estar autenticado en el servidor de *Google*. Si no existieran en su navegador las *cookies* de dicha autenticación, en *4* *Google* mostraría primero una *web* de petición de usuario y contraseña, y después ya se mostraría la página de petición de acceso.

Una vez se ha concedido el permiso, la página de autorización de *Google* redirecciona al usuario

a la página de la aplicación que se especificó como parámetro `next` (*6*). En esta redirección se accede a la página de la aplicación con un *token* temporal como parámetro. el cual sirve, dependiendo del valor que se le hubiese dado al parámetro `session`, para realizar una única petición o para solicitar un *token* de sesión. En el caso de la aplicación, se han solicitado siempre *tokens* de sesión, por lo que el parámetro `session` toma el valor `1`, Estos *tokens* de sesión son almacenados en la base de datos hasta que el usuario inicie una nueva sesión en el servidor o bien decida revocarlos. Por tanto, la solicitud llevada a cabo por la aplicación (*9*) sería la petición de un *token* de sesión, el cual se proporcionaría en *10*.

Debido a que el proceso de obtención de un *token* de sesión es el mismo siempre, se ha decidido implementar este proceso en una página independiente llamada `gforms-gloged`. De esta manera, dicha página puede ser utilizada por todas aquellas páginas que requieran la utilización de *tokens* (`gforms-listforms` y `gforms-listsubmits`). La página `gforms-gloged` solicita el *token* de sesión a partir del *token* temporal y redirecciona a la página que necesita dicho *token*, utilizando el valor que haya recibido como parámetro `go`, de igual manera que lo realiza la página de autorización de *Google* con el parámetro `next`.

La búsqueda del *token* en la base de datos es realizada por el método `gforms::get_token`, el cual también se encarga de la comprobación de su validez. Con este diseño, se separa el código de obtención de *token* de las páginas que lo utilizan, de manera que pueda ser utilizado por todas ellas sin necesidad de reescribir casi código.

En la figura 4.10 se muestra dicho diseño. Cómo se puede ver, es válido para cualquier página que utilice *tokens*, ya sea `gforms-listforms` o `gforms-listsubmits`. Dicha página genérica se representa en el gráfico cómo *lists*.

Cómo se ha comentado anteriormente, en la primera redirección se incluyen como parámetros las direcciones de las *URLs* a las que direccionará tanto la página de *AuthSub* (*Google* en el gráfico) con el parámetro `next`, como `gforms-gloged` (*loged* en el gráfico) con el parámetro `go`.

4.5.2. Obtención de la lista de formularios

La obtención de la lista de formularios es llevada a cabo por la página `gforms-listforms`, cuya finalidad es mostrarle al usuario una lista con los formularios disponibles en su cuenta de *Google Docs*, junto con el texto de cada una de las preguntas de cada uno.

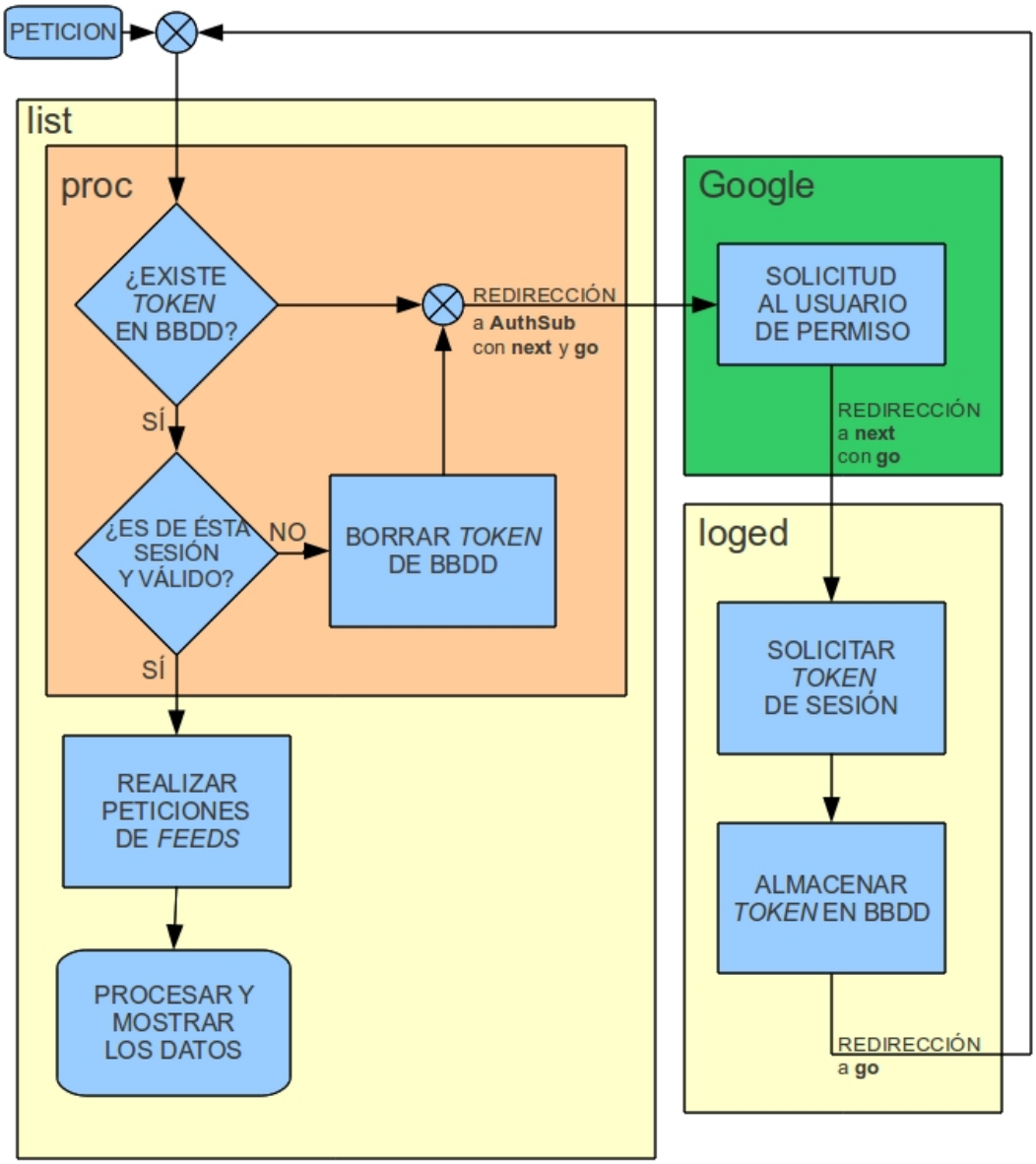


Figura 4.10: Diagrama de flujo del proceso de obtención de token.

Esta lista contiene enlaces que llevan a la página `gforms-edit`, en la cual se añaden nuevos cuestionarios a la aplicación. Cada enlace contiene como parámetros la *URL* del formulario, con su correspondiente clave `formkey`, y el nombre de la pregunta del cuestionario que va a ser utilizada para la identificación. También se añade la opción de no añadir ninguna pregunta por si se quisiera considerar dicho formulario anónimo. Por tanto, existirán tantos enlaces como la suma del número de preguntas de todos los formularios y el número de formularios.

De esta forma, el usuario puede añadir formularios a la aplicación sin necesidad de copiar y pegar ni la *URL* del formulario, ni el campo de texto de la pregunta que va a ser utilizada para la autenticación.

Cómo se comentó en la sección 3.2.1, en versiones anteriores del *API* de *Google Docs*, era posible obtener la lista de formularios existentes en la cuenta de un usuario realizando una petición a la *web* `http://docs.google.com/feeds/documents/private/full/-/form`. Dicha fuente *web* (*'feed'*) dejó de funcionar en una actualización del *API* y no se encuentra disponible actualmente.

Por ello, para obtener la lista de formularios ha sido necesario utilizar la lista de *Spreadsheets* (para la cual sí existe un *'feed'*), y a partir de las claves de cada *Spreadsheet* generar claves de formularios correspondientes a las mismas, en el caso de que existan. Debe recordarse que es posible que exista una *Spreadsheet* sin tener un formulario asociado, pero todos los formularios tienen su correspondiente *Spreadsheet* asociada.

Para obtener la clave de formulario a partir de la clave de su *Spreadsheet* se ha tenido en cuenta el análisis de la estructura de dichas claves descrito en la sección 3.2.1. A partir de la clave de la *Spreadsheet* (`key`) se extraen los últimos 31 caracteres, y se le añaden los 3 caracteres `6MQ`. Estos 34 caracteres constituyen una clave de formulario (`formkey`) válida. Aunque es posible que la clave original de dicho formulario terminara en `A` en lugar de `Q`, es posible acceder igualmente con la clave terminada en `Q`.

Para comprobar que exista un formulario con dicha clave, simplemente se realiza una petición a la URL del formulario (`http://spreadsheets.google.com/viewform?formkey=clave`). Si la página obtenida en respuesta a dicha petición no tuviera ningún formulario significa que no existe un formulario asociado a la *Spreadsheet* a partir de la cual se obtuvo la clave, por lo que se descartaría. Si por el contrario se obtuviera un formulario, se extraerían el nombre de cada una de las preguntas que lo componen, para dar la posibilidad al usuario de que elija cual va a utilizar para identificar a los usuarios que realicen el formulario.

En la figura 4.11 se muestra un diagrama de flujo simplificado del proceso de obtención de la lista de formularios llevado a cabo por la página `gforms-listforms`.

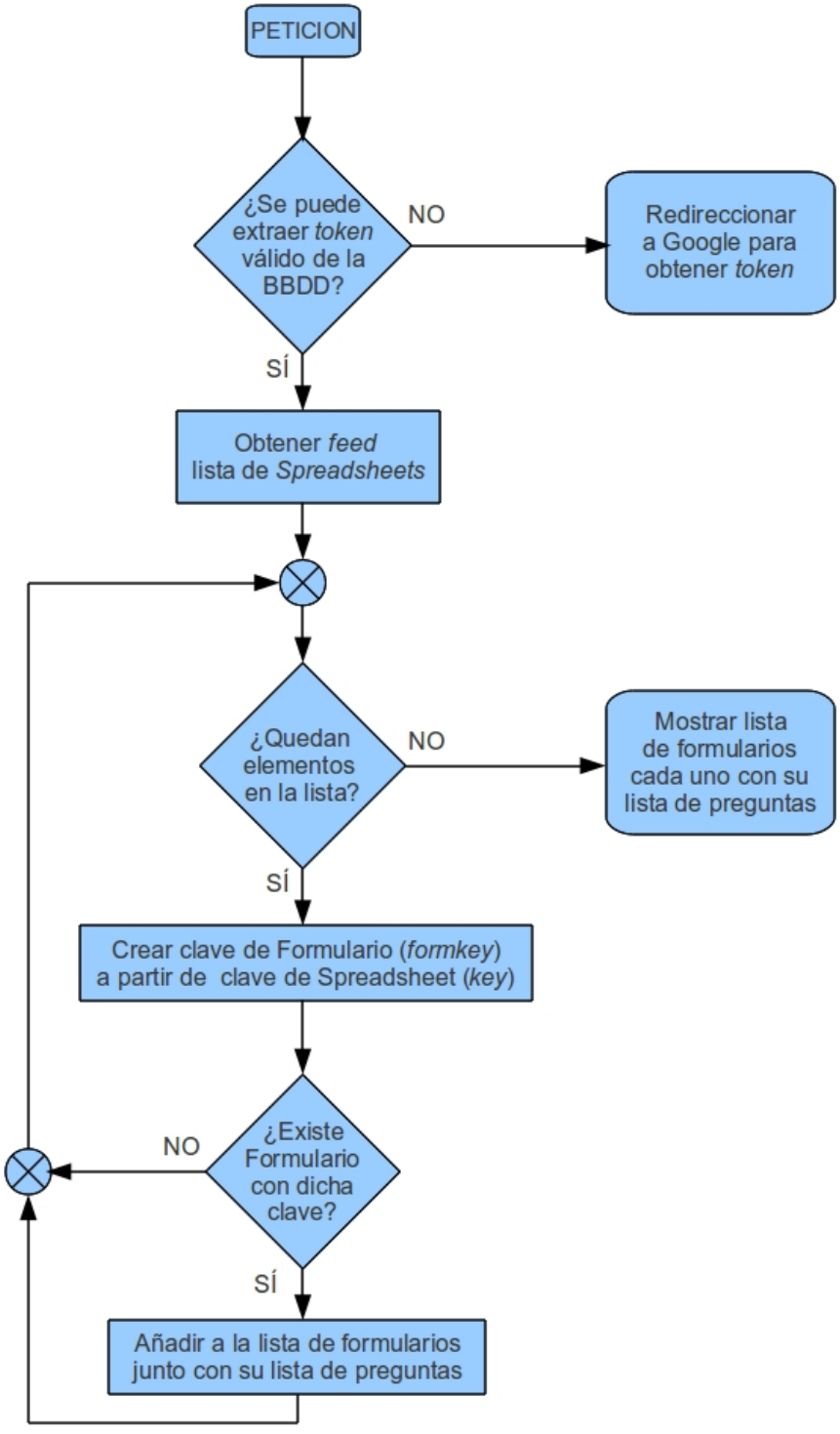


Figura 4.11: Diagrama de flujo del proceso de obtención de lista de formularios.

4.5.3. Obtención de los registros de respuestas y filtrado

El proceso de obtención de los registros (*records*) de la *Spreadsheet* asociada a un Formulario, es decir, las respuestas que se han realizado para dicho Formulario, es llevado a cabo en la página `gforms-listsubmits`.

Aunque existe la posibilidad de que un usuario que descubra la clave pública de acceso al formulario realice respuestas sin utilizar la aplicación *gforms*, dichas respuestas no autenticadas serán filtradas. En dicho caso, existirían en la *Spreadsheet* registros que no tendrían un *hash* válido, es decir, creado por el paquete *gforms* y por tanto existente en la base de datos.

Por ello, el *hash* de cada registro, existente en la *Spreadsheet*, es buscado entre los existentes en la base de datos (tabla `gfanswers` y vista `gfanswersx`) para confirmar la autenticidad del registro. En caso de que se encuentre, se mostrará el registro junto con los datos del usuario al que corresponda dicho *hash*. En caso contrario, será ignorado.

En la figura 4.12 se muestra un diagrama de flujo simplificado del proceso de obtención de las respuestas autenticadas realizado por `gforms-listforms`. Para que dicho diagrama resulte sencillo, no se muestran en detalle algunas operaciones como la obtención de la clave de la *Spreadsheet* a partir de la clave del formulario. Dicha operación consiste en comparar los 31 primeros caracteres de la clave del Formulario (`formkey`) con los 31 últimos caracteres de la clave de cada *Spreadsheet* (`key`) del usuario.

Además de la operación básica de mostrar una lista con todas las respuestas del formulario, se han implementado otras operaciones similares en la misma página `gforms-listsubmits` que reutilizan prácticamente todo su código. El usuario puede acceder a ellas a través de enlaces de esta misma página, los cuales redireccionan a la misma página con un parámetro que se ha denominado `action`, el cual toma el valor de la operación que se quiera llevar a cabo (`showall`, `download` o `filter`). Dichas operaciones pueden ser las siguientes:

Descargar un archivo *CSV* con la lista de respuestas autenticadas

Con esta opción el usuario puede descargarse un archivo en formato *CSV* (*Comma-Separated Values*) que contiene una lista igual a la que se vería en pantalla con todos los registros autenticados.

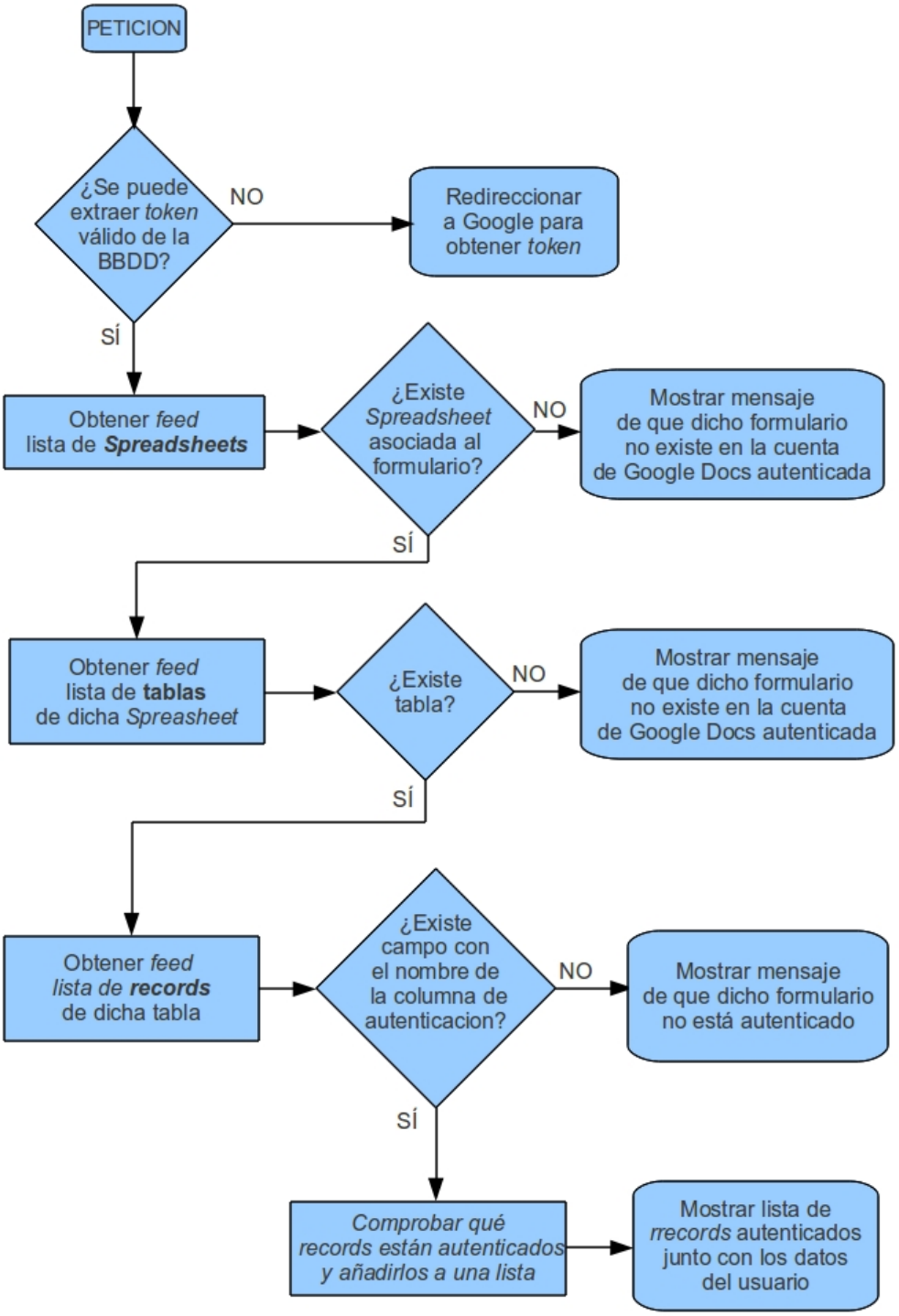


Figura 4.12: Diagrama de flujo simplificado del proceso de obtención de respuestas autenticadas.

Mostrar las respuestas no autenticadas en la misma lista

Esta opción añade a la lista también las respuestas que no están autenticadas, es decir, aquellas que no disponen de un *hash* que se encuentre en la base de datos. Para estas preguntas, la columna del nombre de usuario autenticado figurará como 'No autenticado', de manera que se puedan distinguir claramente de las respuestas que sí están autenticadas.

Eliminar de la base de datos aquellos registros no autenticados

Esta opción elimina de la *Spreadsheet* aquellos registros que no están autenticados, de manera permanente e irreversible. De esta forma, ante un posible ataque es posible 'limpiar' la *Spreadsheet* manteniendo solamente los registros autenticados. Además, al utilizar la opción de ver las estadísticas de las respuestas (*viewanalytics*) se mostrarán las estadísticas únicamente de los formularios autenticados.

Para eliminar cada registro se lleva a cabo la petición de tipo `DELETE` tal como se describe en el *API* de *Google Spreadsheet*. Una vez efectuada dicha petición, las listas de *feeds* obtenidas anteriormente dejan de ser válidas, ya que cada registro cambia su identificador respecto a cuando se obtuvo, de manera que no es posible realizar más operaciones utilizando dichos identificadores. Por ello, cada vez que se borra un registro, se reutiliza todo el código anterior, volviendo a realizar las peticiones de *feeds*, para así borrar el resto de los posibles registros no autenticados que puedan seguir existiendo en la *Spreadsheet*.

4.6. Manejo de formularios en la aplicación

En esta sección se explican cómo se realizan las funcionalidades de la aplicación que permiten añadir, editar y eliminar los formularios.

4.6.1. Creación y edición de formularios

Los procesos de creación y de edición de formularios son llevados a cabo por la página `gforms-edit`. Si el proceso es el de edición, esta página recibe un parámetro denominado `item_id` con el valor del identificador del objeto que se desea editar. En caso de creación, no recibirá dicho parámetro. De esta manera, la página puede distinguir entre el tipo de proceso a realizar, reutilizándose la mayoría del código para ambos casos.

La página `gforms-edit` muestra los campos de los parámetros de un formulario: título, descripción, *URL* (de la cual se extrae la `formkey` del formulario), texto de la pregunta en la cual se almacena el *hash*, fecha inicial, fecha final y número de intentos. En el caso de edición de un formulario, estos campos aparecerán con el contenido del formulario a editar, y el campo de *URL* deshabilitado, de manera que no se pueda modificar (si se cambia la *URL* el formulario sería distinto).

En el caso de crear un nuevo formulario, se mostrarían todos los campos vacíos, a menos que se hubiera accedido a dicha página seleccionando el formulario de la lista (ver sección 4.5.2), en cuyo caso ya aparecerían el campo de la *URL* y el del texto de la pregunta. Este último excepto si se hubiera elegido el formulario como anónimo, en cuyo caso permanecería vacío.

Para la creación de la página `gforms-edit` se ha utilizado la plantilla del *OpenACS Template System* denominada *adform*. Una vez el usuario ha introducido todos los campos, se llevan a cabo las siguientes comprobaciones:

- Que la *URL* sea correcta y se corresponda con un formulario.
- Que no exista ningún formulario en la base de datos con dicha *URL*.
- Que la fecha límite sea posterior a la de inicio.
- Que la fecha límite sea posterior a la actual.
- Que el número de intentos sea un número entero mayor que cero.

Si todos los datos introducidos son válidos, el formulario es añadido a la base de datos, o bien es creada una nueva revisión en el caso de que el proceso sea de edición.

4.6.2. Eliminación de formularios

Para eliminar los formularios se ejecuta el *script* `gforms-delete.tcl`, el cual no tiene ninguna página *ADP* asociada, por lo que una vez realizado el proceso, redirecciona a la página principal.

Al eliminar un formulario de la base de datos, se eliminan también todos los elementos de la base de datos correspondientes a las respuestas de dicho formulario (tabla `gfanswers`).

4.7. Internacionalización

Los textos del paquete desarrollado se han localizado a los idiomas Español e Inglés utilizando el paquete de internacionalización proporcionado por *Open ACS*. Este paquete permite no sólo que las aplicaciones estén disponibles en varios idiomas, encargándose de mostrar aquel en que está configurada la aplicación, sino que además ofrece la posibilidad de que otras personas sin ninguna relación con el creador de la aplicación lleven a cabo la localización a otros idiomas sin necesidad de modificar el código del programa.

Para que esto sea posible, en todos los archivos *TCL* y *ADP* del paquete, en lugar de introducir textos se han utilizado etiquetas con una clave única para cada texto. La estructura de dichas etiquetas para los ficheros *ADP* es la siguiente:

```
#gforms.Key#
```

En el caso de los ficheros *TCL*, se utiliza un proceso definido por el paquete de internacionalización, denotado únicamente por el símbolo del guión bajo ('_') y cuyo atributo es la clave del texto, tal y cómo se puede ver a continuación:

```
[_ gforms.Key]
```

Para cada idioma se crea un archivo *XML* con todos los textos del paquete correspondientes a cada clave. Estos archivos se encuentran en la carpeta `catalog` (ver sección 4.2). La estructura de dichos ficheros se muestra a continuación:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<message_catalog package_key="gforms" locale="es_ES" charset="ISO-8859-1">
  <msg key="Add_form">Añadir nuevo formulario</msg>
  <msg key="Description">Descripción</msg>
  <msg key="Name">Nombre</msg>
...
</message_catalog>
```

4.8. Posible violación de las condiciones del servicio de *Google*

La no existencia de *API* para *Google Formularios* ha supuesto la desventaja de tener que analizar el funcionamiento de la interfaz de usuario proporcionada y tener que adaptar la aplicación a las limitaciones que ésta ofrece. Pero también conlleva una posible violación de las condiciones del servicio de *Google*[2] al realizar el acceso a *Google Formularios* a través del servidor en lugar de a través del navegador del usuario, como se explicó en la sección 4.4.

La cláusula de las condiciones que trata sobre este tema es la siguiente:

> 5.3 Usted se compromete a no acceder ni a intentar acceder a ninguno de los Servicios por ningún otro medio que no sea la interfaz provista por Google, a menos que así se haya establecido en un acuerdo por escrito independiente entre usted y Google. Se compromete en concreto a no acceder, o intentar acceder, a ninguno de los Servicios utilizando medios automatizados, incluidos secuencias de comandos o rastreadores *web*, y se asegurará de cumplir las instrucciones dispuestas en cualquier archivo robots.txt correspondiente a los Servicios.

Se hace especial hincapié en el uso de rastreadores robots, lo cual parece referirse a aquellos medios automatizados que realizan secuencias que conllevan una gran cantidad de peticiones. En la aplicación se puede considerar que no se llevan a cabo este tipo de secuencias, ya que simplemente se realizan peticiones cuando un cliente lo solicita, de igual manera que lo realizaría a través de su interfaz. Además, deja la puerta abierta a establecer un acuerdo con *Google*.

En la descripción del paquete implementado se ha comentado este hecho, advirtiendo que su utilización en plataformas comerciales sin la autorización de *Google* podría conllevar problemas legales.

La utilización de la aplicación durante su desarrollo no se considera que pueda acarrear problemas al no tratarse de una explotación comercial. Además, por parte de *Google* sería difícil localizar y distinguir las peticiones realizadas por la aplicación de las que realizaría un navegador. En el caso de la realización de peticiones masivas robotizadas, sí sería detectable, y es lo que *Google* prohíbe explícitamente.

[2] http://www.google.com/accounts/TOS

Capítulo 5

Conclusión y líneas futuras

En este capítulo se exponen tanto las conclusiones a las que se ha llegado después de realizar el trabajo, cómo las posibles ampliaciones del proyecto que podrían llevarse a cabo.

5.1. Conclusión

Este proyecto tenía como objetivos integrar el servicio de realización de cuestionarios *Google Formularios* dentro de la plataforma *.LRN*, conseguir una identificación segura de los usuarios, poder filtrar las respuestas que no procedan de usuarios de la plataforma y permitir el manejo de la información desde dentro de la aplicación. A pesar de la dificultad que ofrece desarrollar un paquete que integre un servicio que no está diseñado para ello (*Google Formularios*), se ha conseguido dicho objetivo.

Durante el desarrollo del proyecto ha sido necesario aprender, en primer lugar a programar aplicaciones (paquetes) para *.LRN* y en consecuencia para *OpenACS*, lo cual incluye módulos cómo el sistema de plantillas y el sistema de objetos de la base de datos del *Content Repository*. En segundo lugar, ha sido necesario conocer un nuevo lenguaje de programación basado en *script* (*TCL*) y saber utilizar el *API* de un sitio web para interactuar con él a través de peticiones *HTTP*. También ha sido necesario llevar a cabo el diseño completo de una aplicación, y ser capaz de buscar herramientas que permitan llevar a cabo las funciones que ésta requiera (módulo *SSL* para *.LRN* y librería *TclCurl*).

La limitación de no disponer de un *API* de *Google Formularios* supone que la aplicación no pueda encargarse de añadir dicho campo, ya que para ello sería necesario realizar un análisis de

ingeniería inversa del código *Javascript* de la página de modificar formularios, tarea de una gran complejidad. Dicho problema se ha solucionado obligando al usuario administrador a reservar un campo del formulario para llevar a cabo la identificación. Se hubiera preferido realizar la aplicación de manera que esto no fuera necesario, pero se consideró que ante las limitaciones existentes, era la mejor decisión posible para el diseño.

En resumen, se ha conseguido que la aplicación *Google Forms* quede integrada en *.LRN* permitiendo identificar a los usuarios. Para ello se han investigado distintas posibilidades de diseño y se ha implementado la que se ha considerado idónea dadas las limitaciones. Por último, se ha verificado que la aplicación cumpla con todas sus funcionalidades.

5.2. Trabajos futuros

El principal riesgo que supone utilizar herramientas externas, sobre todo en el caso de que no ofrezcan oficialmente la posibilidad de integración, son los posibles cambios que puedan producirse en dichas herramientas. Una modificación en el código de los formularios ofrecidos por *Google* podría suponer un mal funcionamiento de la aplicación. Si se produjera dicho evento, sería necesario volver a analizar la nueva estructura de dicho código y, diseñar e implementar las modificaciones correspondientes en la aplicación. Para el caso de *Google Spreadsheets*, a pesar de existir un *API* también existe la posibilidad de que se introduzcan variaciones, por lo que sería necesario modificar igualmente la aplicación, aunque en este caso sería más sencillo al no requerir un análisis de código *HTML* sino simplemente consultar un nuevo *API*. En definitiva, una aplicación de esta naturaleza requiere un proceso de mantenimiento para adaptarse a la evolución de los servicios con los que interacciona.

Una funcionalidad que se le podría añadir a la aplicación sería la de soportar formularios con varias páginas. El diseño actual de la aplicación permite que dichos formularios sean utilizados sin producirse errores tratándolos cómo formularios anónimos, es decir, no es posible llevar a cabo el proceso de autenticación de los usuarios. Estos formularios tienen una estructura distinta, que sería necesario analizar en detalle, y llevar a cabo un nuevo diseño del proceso de realización de cuestionarios, que podría ser considerablemente distinto al llevado a cabo con la realización de las páginas `gforms-viewform` y `gforms-response`.

Otra funcionalidad que se podría desarrollar sería la de compartir formularios entre comunidades. Esto resultaría especialmente sencillo cuando estas pertenecen a una misma plataforma,

y por tanto dichas comunidades comparten los datos en una misma base de datos. Pero si dichas comunidades pertenecieran a distintas plataformas, los servidores de éstas tendrían que interaccionar para conocer las respuestas realizadas en otras comunidades y no filtrarlas.

APÉNDICES

APÉNDICE A

Planificación y presupuesto

Este apéndice consta de dos secciones. En la primera, se muestra una planificación del tiempo dedicado a cada una de las tareas, y en la siguiente, el presupuesto del proyecto considerando costes personales y materiales. Tanto el tiempo cómo el coste son calculados de manera aproximada.

A.1. Planificación

En la tabla A.1 se muestra el tiempo aproximado que se ha dedicado a cada una de las tareas en las que se puede descomponer el proyecto.

Tabla A.1: Planificación de tareas.

Tarea	***Duración***
Elaboración de la especificación de requisitos.	*1 semana*
Estudio del desarrollo de paquetes para *.LRN*. Implementación de un paquete sencillo que muestre variables del código *TCL* en la página *ADP*, incluyendo código *HTML*.	*3 semanas*
Estudio del desarrollo de paquetes para *.LRN*. Implementación de un paquete sencillo que permita añadir y modificar formularios (interfaz y *BBDD*) utilizando el *Content Repository* y el sistema de plantillas.	*5 semanas*
Continúa en la siguiente página	

Tabla A.1 – continúa de la página anterior

Tarea	***Duración***
Estudio del desarrollo de paquetes para *.LRN*. Implementación en un paquete sencillo que permita añadir a la *BBDD* respuestas de los formularios mediante la herencia del *Content Repository*.	*2 semanas*
Análisis de las posibilidades de *Google Formularios* y diseño de un modelo de autenticación (utilizando un *hash*).	*3 semanas*
Estudio del análisis sintáctico (*parser*) de *XML* y *HTML* (que no cumpla *XML*) en *.LRN* (*tDOM*) e implementación de obtención y modificación de código *HTML*.	*2 semanas*
Análisis de la utilización de conexiones seguras (*TLS/SSL*) en *.LRN* y realización de peticiones *HTTPS*.	*2 semanas*
Análisis de la utilización de *TclCurl* para realizar peticiones a *Google Spreadsheets*. Obtención correcta de *token* y realización de peticiones de *feeds*.	*2 semanas*
Diseño definitivo de la aplicación, tanto base de datos cómo estructura de código.	*1 semana*
Implementación definitiva de la aplicación, con todas sus funciones e interfaces. Realización de formularios.	*2 semanas*
Implementación definitiva de la aplicación, con todas sus funciones e interfaces. Autenticación en *Google* y creación de lista de formularios.	*2 semanas*
Implementación definitiva de la aplicación, con todas sus funciones e interfaces. Obtención del registro de respuestas y filtrado.	*2 semanas*
Realización de pruebas y corrección de errores.	*2 semanas*
Redacción del estado del arte y las tecnologías implicadas.	*2 semanas*
Redacción del desarrollo del proyecto.	*2 semanas*
Redacción del resto de secciones de la memoria del proyecto.	*2 semanas*
Revisión de la memoria del proyecto.	*1 semana*
TOTAL	**36 semanas**

A.2. Presupuesto

En esta sección se presentan justificados los costes globales de la realización de este Proyecto Fin de Carrera. Tales costes, imputables a gastos de personal y de material, se pueden deducir de las tablas A.3 y A.4.

En primer lugar se muestra en la tabla A.2 el tiempo aproximado dedicado a cada uno de los objetivos especificados en la sección 1.2. Se podría decir que dichos objetivos corresponden a distintas fases, sin embargo, al entrelazarse el desarrollo de *software* con el aprendizaje de las plataformas implicadas, cómo se vio en la tabla A.1, estas fases no se desarrollaron de manera secuencial, por lo que es preferible hablar de objetivos en lugar de fases.

Objetivo 1	*Documentación y aprendizaje de las plataformas implicadas*	250 *horas*
Objetivo 2	*Diseño del software*	20 *horas*
Objetivo 3	*Desarrollo del software*	180 *horas*
Objetivo 4	*Realización de pruebas y corrección de errores*	50 *horas*
Objetivo 5	*Redacción de la memoria del proyecto*	150 *horas*
TOTAL		**650 *horas***

Tabla A.2: Tiempo aproximado dedicado a cada objetivo del Proyecto.

De la la tabla A.2 se desprende que el tiempo total dedicado por el proyectando ha sido de en torno a 650 horas, de las cuales aproximadamente unas 50 han sido compartidas con el tutor del proyecto. Considerando el respectivo salario en función de la categoría del ingeniero, los costes de personal se recogen en la tabla A.3

Ingeniero	**Categoría**	**Dedicación**	**Coste**
Abelardo Pardo Sánchez	Ingeniero senior (35 €/hora)	50 horas	1.750 €
Juan González Adrados	Ingeniero (20 €/hora)	650 horas	13.000 €
TOTAL			**14.750 €**

Tabla A.3: Costes del personal.

Ordenador de gama media	600 €
Software	0 €
Local (durante 10 meses, con un coste de 120 €/mes)	1.200 €
Gastos varios	200 €
TOTAL	**2.000 €**

Tabla A.4: *Costes de material.*

En la tabla A.4 se recogen los costes de material desglosados en equipo informático, *software*, local de trabajo y otros gastos. Ni el propio sistema, ni la máquina virtual tienen requisitos considerables, por lo que un ordenador de gama media resulta suficiente para llevar a cabo el Proyecto. Cómo se describió en el capítulo 3, todo el *software* utilizado es gratuito. Dicho *software* se ejecutó sobre el sistema operativo, también gratuito, *Ubuntu*. Para la redacción de la memoria también se han utilizado herramientas gratuitas, *Kile* cómo editor de LaTeX, y *OpenOffice Draw* para la creación de imágenes. Los costes de electricidad y acceso a internet se incluyen en el coste del local. El resto de gastos, tales cómo impresión de documentos y desplazamientos se incluyen en el apartado 'Gastos varios'.

Concepto	**Importe**
Costes personal	14.750 €
Costes material	2.000 €
Base imponible	16.750 €
I.V.A. (18 %)	3.015 €
TOTAL	**19.765 €**

Tabla A.5: Presupuesto.

A partir de estos datos, el presupuesto total asciende a 19.765 €, incluyendo el I.V.A., tal cómo es mostrado en la tabla A.5.

APÉNDICE B

Manual de instalación

Este apéndice constituye una guía para que el administrador pueda instalar la aplicación en la plataforma *.LRN*. Se presupone que dicha plataforma ya se encuentra instalada y configurada en su instalación básica. Esta instalación se centra en la utilización de *AolServer4* cómo servidor y *Linux* cómo sistema operativo. No obstante, se indica para otros sistemas operativos dónde pueden obtenerse los paquetes, aunque no se detalle cómo realizar la instalación de estos. Las rutas de archivos se corresponden para la instalación utilizada por la máquina virtual proporcionada por la Universidad Carlos III de Madrid [1], pudiendo diferir en otras instalaciones.

B.1. Instalación de paquetes requeridos

Para el correcto funcionamiento de la aplicación es necesario instalar dos paquetes no incluidos en la instalación básica de *.LRN*. Estos paquetes son *nsopenssl*, el cual permite realizar conexiones seguras (*SSL*), y *TclCurl*, el cual posibilita a la aplicación llevar a cabo peticiones *HTTPS* gracias al anterior.

B.1.1. Instalación de *nsopenssl*

Para utilizar *nsopenssl*, en primer lugar es necesario instalar *OpenSSL*. Este paquete está disponible para distintos sistemas operativos en el siguiente enlace.

http://sourceforge.net/projects/tls/files/

[1] https://gradient.it.uc3m.es/xowiki/dotlrn_vm

Para el caso de *AOLserver4* en *Linux*, los archivos deben de situarse en la siguiente ruta:

```
/usr/lib/aolserver4/lib/tcllib1.10/tls/
```

Una vez instalado *OpenSSL*, se procede a instalar *nsopenssl*. El paquete se puede descargar del siguiente enlace.

http://aolserver.cvs.sourceforge.net/viewvc/aolserver/nsopenssl/

Después de su correcta instalación[2], si estamos utilizando el *AOLserver4* en *Linux*, deberá aparecer el archivo *nsopenssl.so* en las siguientes ubicaciones.

```
/usr/lib/aolserver4/lib/
/usr/lib/aolserver4/bin/
```

Para el correcto funcionamiento de *nsopenssl* es necesaria la versión *0.9.7* de *libssl* y *libcrypto*, por lo que si disponemos de dichas librerías en una version posterior, deberemos crear enlaces simbólicos o directamente renombrar una copia para que tenga dicho nombre.

```
cp /usr/lib/libssl.so.0.9.8 /usr/lib/libssl.so.0.9.7
cp /usr/lib/libcrypto.so.0.9.8 /usr/lib/libcrypto.so.0.9.7
```

Por último, para utilizar *SSL* es necesario disponer de una clave privada. Para ello, utilizaremos el siguiente *script*[3].

```
perl /usr/lib/ssl/misc/CA -newcert
```

Esto generará los dos archivos que requiere la clave:

```
aolserver/dotlrn/etc/certs/certfile.pem
aolserver/dotlrn/etc/certs/keyfile.pem
```

Una vez realizada cada instalación, es necesario reiniciar el sistema.

[2]Para realizar la instalación puede seguirse la guía disponible en http://www.openacs.org/doc/install-nsopenssl.html

[3]Existe una guía más detalla para realizar este proceso en http://www.openacs.org/doc/install-ssl.html

B.1.2. Instalación de *TclCurl*

Existen distintas versiones del paquete *TclCurl* dependiendo del sistema operativo. Todas pueden descargarse en el siguiente enlace, junto con sus correspondientes manuales de instalación:

http://personal5.iddeo.es/andresgarci/tclcurl/english/download.html

Para el caso de *Linux*, no necesitamos realizar dicha descarga, pudiendo descargarlo e instalarlo directamente utilizando los repositorios. Para ello debemos de ejecutar en la consola el siguiente comando.

```
sudo apt-get install tclcurl
```

Si la instalación se ha producido correctamente, los archivos del paquete aparecerán instalados en la siguiente ruta.

```
/usr/lib/TclCurl7.16.2/
```

B.2. Instalación de la aplicación

El proceso de instalación del paquete *gforms* en una comunidad de la plataforma *.LRN* lleva dos fases. En la primera se instala el paquete en la plataforma para que pueda estar disponible en cualquier comunidad. En la segunda fase, se realiza una instancia de dicho paquete en la comunidad que se desee utilizar.

B.2.1. Instalación del paquete *gforms*

Para instalar el paquete *gforms*, en primer lugar deberemos situar la carpeta con todos sus archivos en la carpeta *packages*.

```
aolserver/dotlrn/packages/gforms/
```

A continuación deberemos utilizar el navegador para acceder al portal de *.LRN* cómo usuario administrador. Una vez identificados cómo administradores, debemos acceder a la página del *Package Manager*. Podemos acceder a dicha página introduciendo directamente su *URL*:

http://SERVIDOR/acs-admin/apm/

O navegando entre las distintas páginas de administración que se exponen en las figuras B.1, B.2, B.3 y B.4.

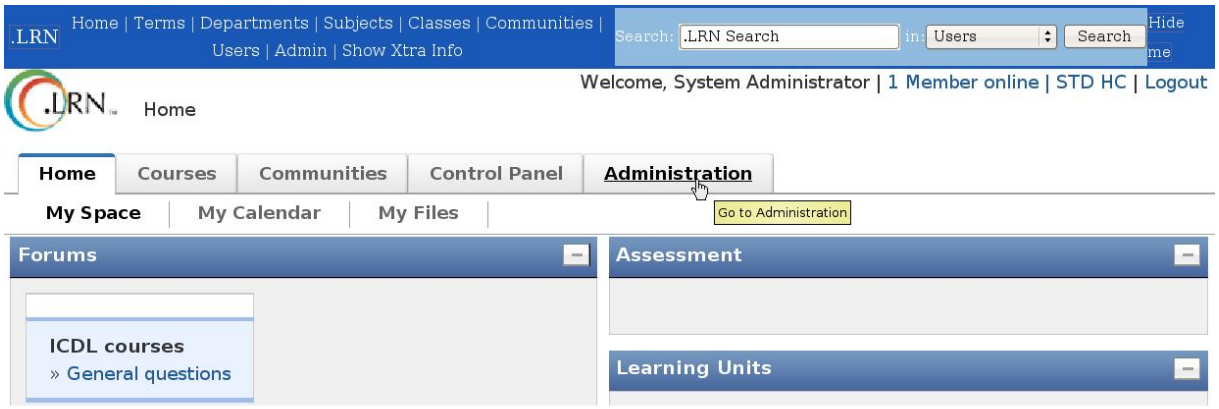


Figura B.1: Página principal.

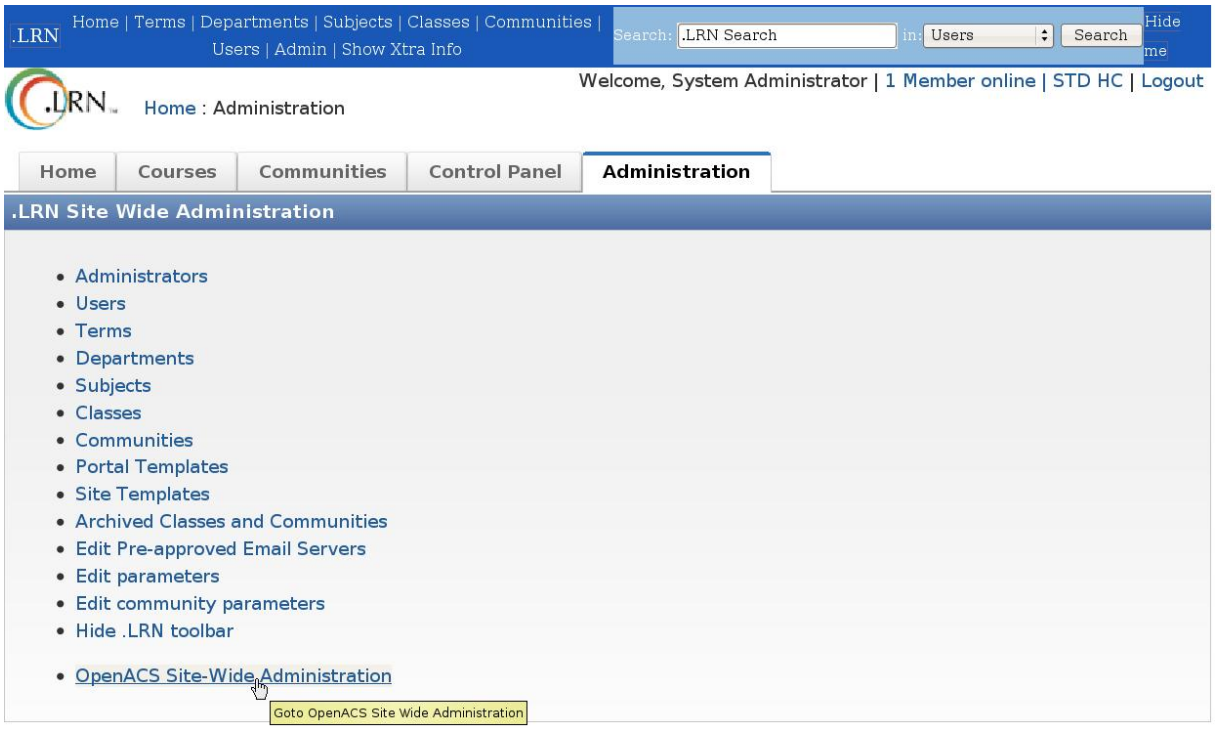


Figura B.2: Página de administración.

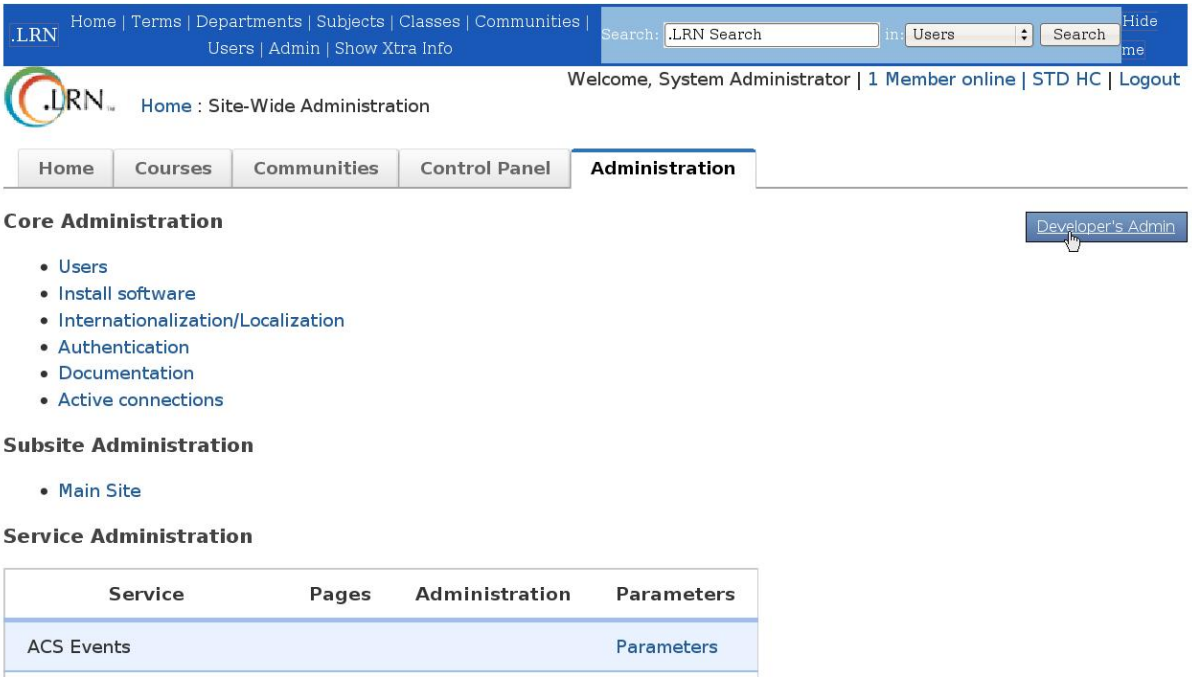


Figura B.3: Página de administración de *OpenACS*.

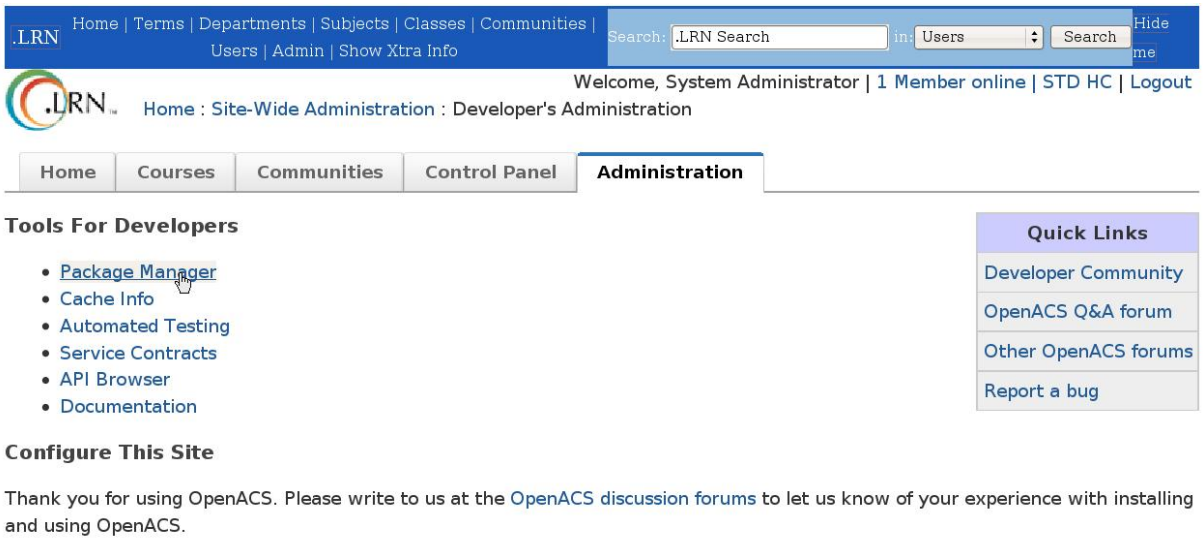


Figura B.4: Página de administración para desarrolladores.

Una vez en el *Package Manager*, deberemos pulsar sobre la opción de instalar nuevos paquetes situada en la parte inferior de la página, cómo muestra la B.5.

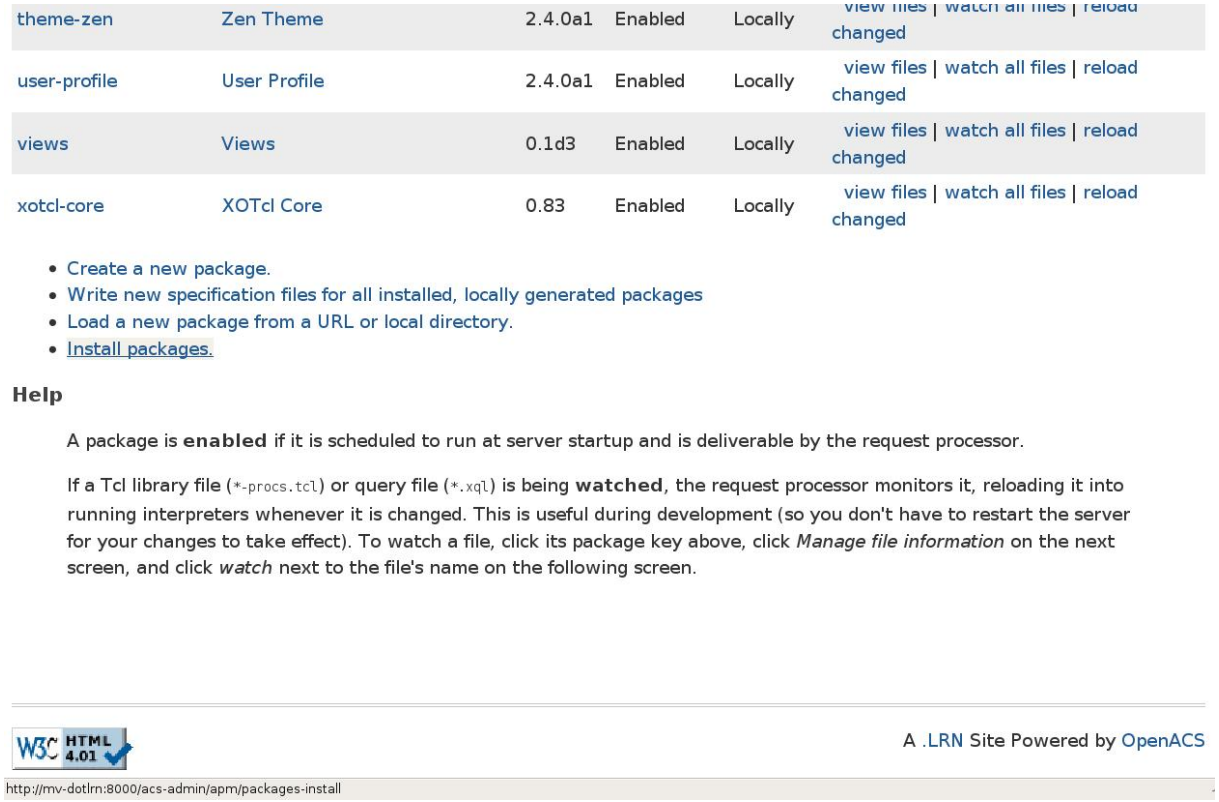
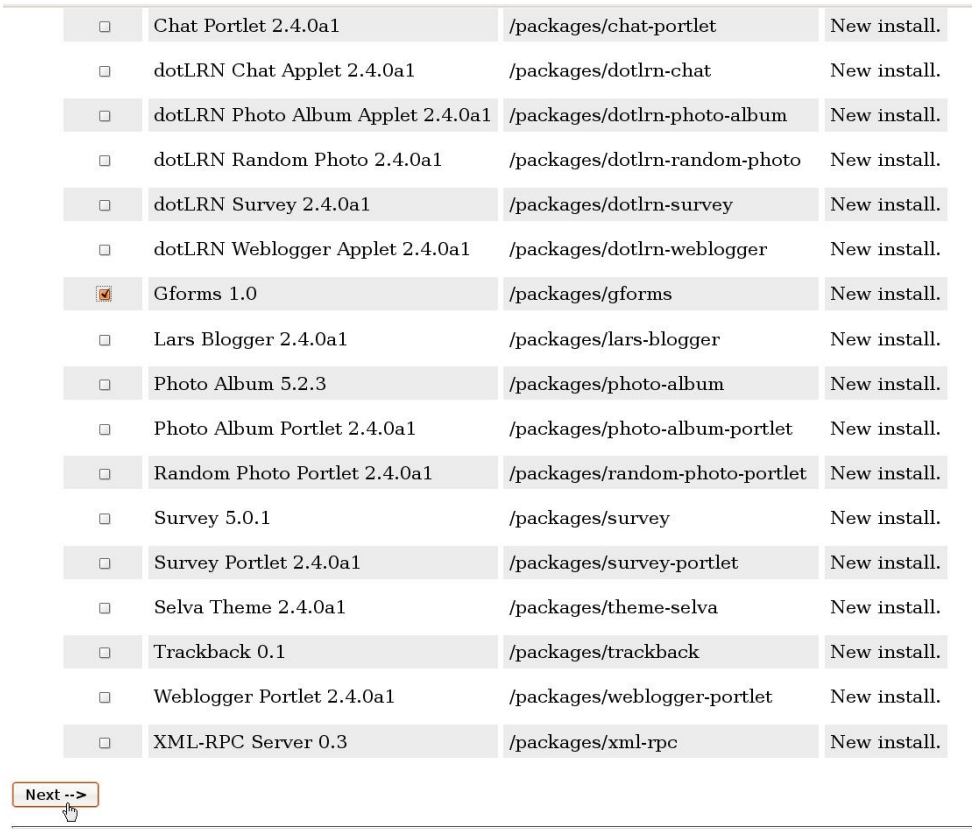


Figura B.5: *Package Manager*.

A continuación seleccionamos el paquete *Gforms* y pulsamos el correspondiente botón de instalar, cómo aparece en la figura B.6.

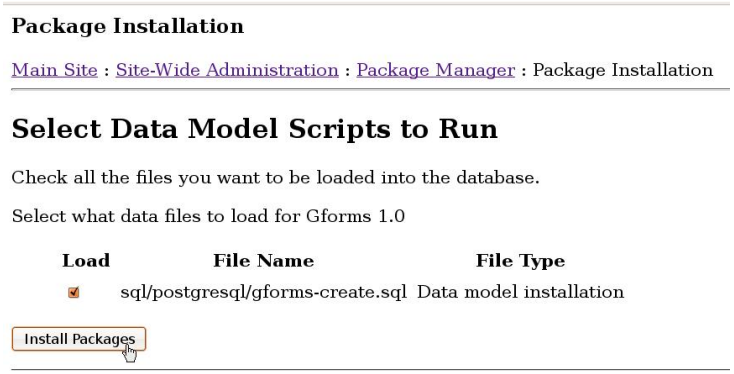


☐	Chat Portlet 2.4.0a1	/packages/chat-portlet	New install.
☐	dotLRN Chat Applet 2.4.0a1	/packages/dotlrn-chat	New install.
☐	dotLRN Photo Album Applet 2.4.0a1	/packages/dotlrn-photo-album	New install.
☐	dotLRN Random Photo 2.4.0a1	/packages/dotlrn-random-photo	New install.
☐	dotLRN Survey 2.4.0a1	/packages/dotlrn-survey	New install.
☐	dotLRN Weblogger Applet 2.4.0a1	/packages/dotlrn-weblogger	New install.
☑	Gforms 1.0	/packages/gforms	New install.
☐	Lars Blogger 2.4.0a1	/packages/lars-blogger	New install.
☐	Photo Album 5.2.3	/packages/photo-album	New install.
☐	Photo Album Portlet 2.4.0a1	/packages/photo-album-portlet	New install.
☐	Random Photo Portlet 2.4.0a1	/packages/random-photo-portlet	New install.
☐	Survey 5.0.1	/packages/survey	New install.
☐	Survey Portlet 2.4.0a1	/packages/survey-portlet	New install.
☐	Selva Theme 2.4.0a1	/packages/theme-selva	New install.
☐	Trackback 0.1	/packages/trackback	New install.
☐	Weblogger Portlet 2.4.0a1	/packages/weblogger-portlet	New install.
☐	XML-RPC Server 0.3	/packages/xml-rpc	New install.

Next -->

Figura B.6: Selección del paquete a instalar.

Debemos seleccionar la opción de ejecutar el *script* de creación de tablas de la base de datos, cómo puede verse en la figura B.7..



Package Installation

Main Site : Site-Wide Administration : Package Manager : Package Installation

Select Data Model Scripts to Run

Check all the files you want to be loaded into the database.

Select what data files to load for Gforms 1.0

Load	File Name	File Type
☑	sql/postgresql/gforms-create.sql	Data model installation

Install Packages

Figura B.7: Ejecución de *script* de base de datos.

Si la instalación se ha producido correctamente, aparecerá la pantalla de la figura B.8.

Figura B.8: Paquete instalado con éxito.

Por último, debemos reiniciar el servidor para que esté disponible el paquete.

B.2.2. Instanciación del paquete en la comunidad

Una vez instalado el paquete en la plataforma, debemos instanciarlo en cada comunidad en que lo deseemos utilizar. Para ello, debemos ir a la página de *Site Map*, a la cual podemos llegar introduciendo directamente su *URL*:

`http://SERVIDOR/admin/site-map/`

O bien, navegando entre las distintas páginas de administración, en cuyo caso en primer lugar deberemos repetir los pasos de las figuras B.1 y B.2, para después pulsar en el enlace *Main site* cómo se muestra en la figura B.9.

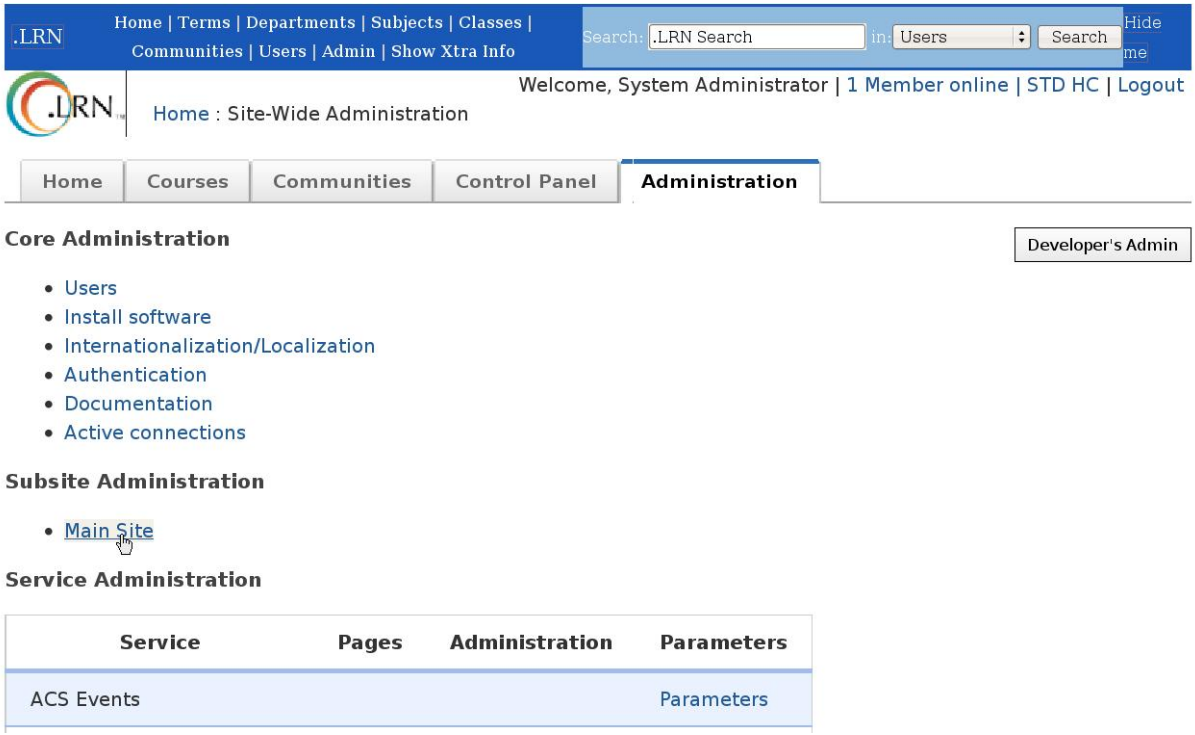


Figura B.9: Página de administración de *OpenACS*.

En la página de administración general se encuentra el enlace al *Site Map* cómo muestra la figura B.10.

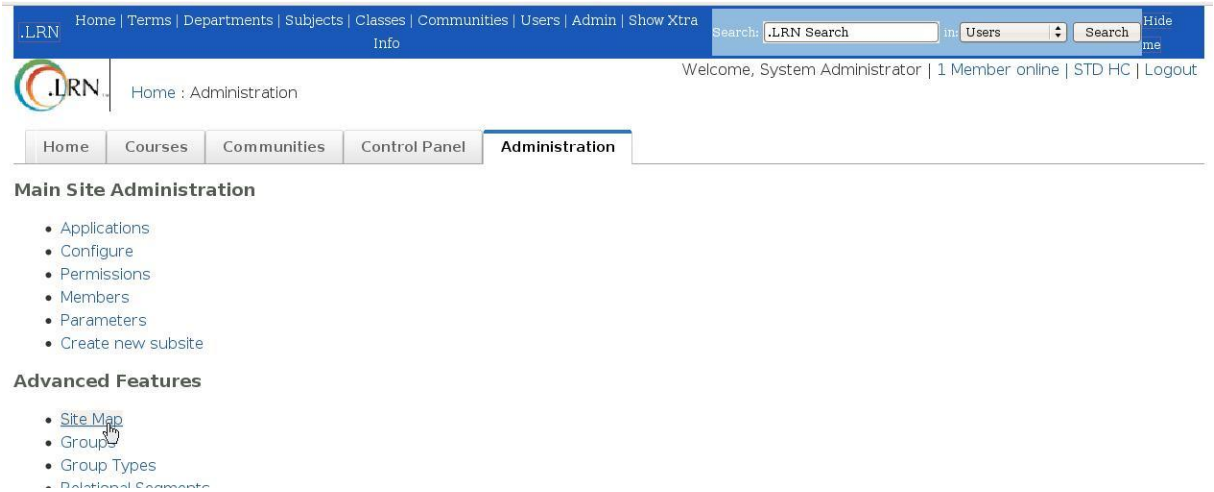


Figura B.10: Página de administración general.

En el *Site Map* debemos de localizar la comunidad en la cual queremos instanciar el paquete. Las comunidades se encuentran dentro de la carpeta *clubs*, la cual se ubica dentro de la carpeta *dotlrn*. En la figura B.11 se muestra cómo ejemplo una comunidad llamada *newcom*.

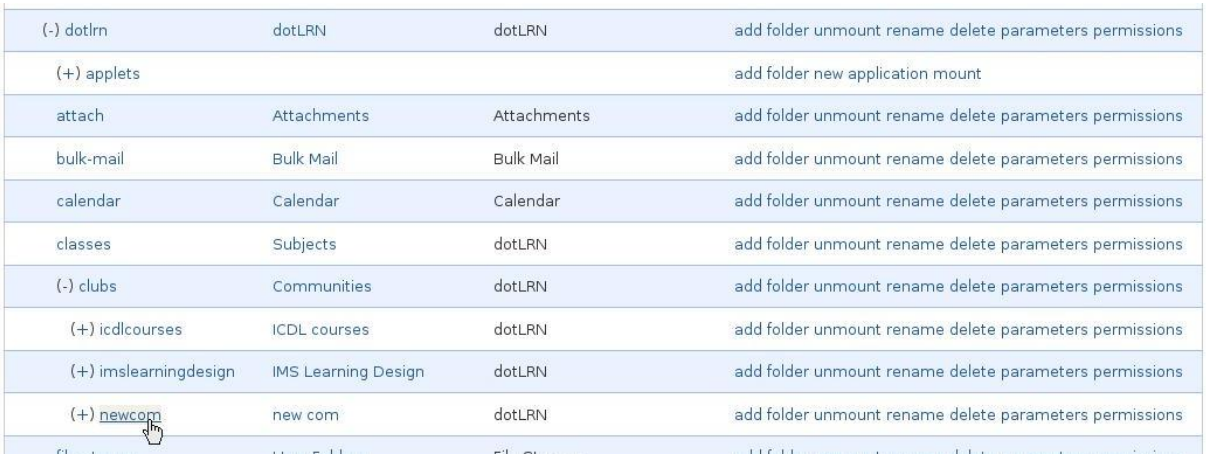


Figura B.11: *Site Map*.

Por último debemos de seleccionar de la lista de paquetes *Gforms*, escribir el nombre de la instancia y montar el paquete, cómo muestra la figura B.12.

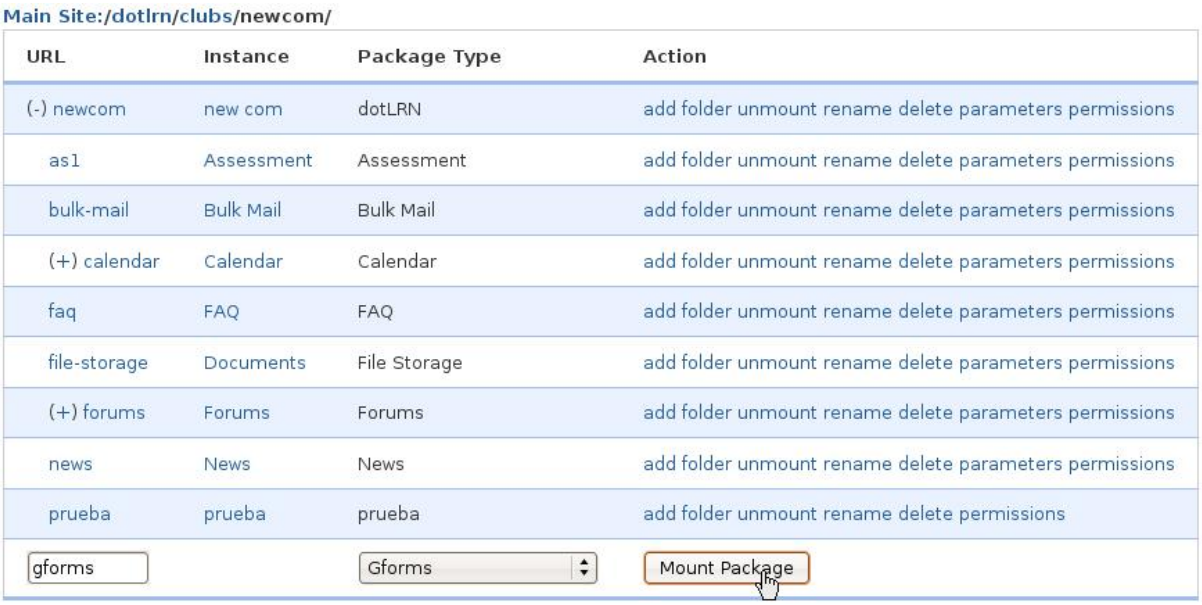


Figura B.12: Instanciación del paquete.

De esta forma, ya estará instanciado el paquete, el cual tendrá su propia *URL* que deberemos enlazar cómo deseemos desde nuestra comunidad, para que los usuarios puedan acceder. En el ejemplo de las figuras anteriores, la *URL* sería:

`http://SERVIDOR/dotlrn/clubs/newcom/gforms/`

APÉNDICE C

Manual de usuario

El objetivo de este apéndice es indicar a los usuarios cómo acceder a todas las funcionalidades ofrecidas por la aplicación. Se divide en dos partes, una para usuarios sin permisos de administración y otra para usuarios con permisos de administración. Cabe recordar que los usuarios con permisos de administración también pueden acceder a la funcionalidad ofrecida a los usuarios sin permisos (realizar cuestionarios).

C.1. Usuarios sin permisos de administración

Cómo se explicó en la sección 4.1.3, los usuarios sin permisos de administración únicamente pueden ver la lista de formularios y realizar éstos (si cumplen los requisitos) a través de la aplicación, quedando registrada su identidad.

C.1.1. Lista de formularios

En la página principal de la aplicación se muestra la lista de formularios disponibles en la comunidad junto con todos sus parámetros.

Name	Description	Start	Deadline	Tries
Vaca	Cuenta lo que quieras sobre este animal	2010-10-12 17:05:00	2011-01-17 15:05:00	0/2
Nuevo formulario	Responde a una serie de preguntas basicas.	2010-10-12 19:45:00	2011-01-01 00:00:00	0/1

Figura C.1: Página principal para usuarios no administradores.

C.1.2. Realizar formulario

Para realizar un formulario, tan sólo hay que pulsar sobre el enlace existente sobre su nombre en la lista de formularios. A continuación aparecerá el formulario (siempre y cuando exista en el servidor de *Google* y se cumplan los requisitos de plazo e intentos), tal cómo muestra de ejemplo de la figura C.2.

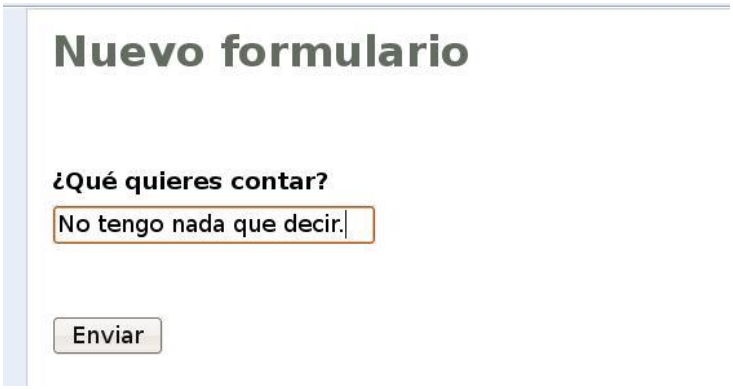


Figura C.2: Ejemplo de formulario.

Si el formulario tuviera campos obligatorios y estos no se hubieran cumplimentado, el formulario volvería a aparecer, no registrándose las respuestas hasta que no se enviara con todos los campos.

C.2. Usuarios con permisos de administración

Los usuarios con permisos de administración tienen acceso a todas las funcionalidades de la aplicación.

C.2.1. Lista de formularios

En este caso, la página principal muestra la lista de formularios con cuatro enlaces para cada cuestionario (además del de su realización), tres enlaces en la parte superior para la realización de cuestionarios, y en el caso de que el usuario haya permitido el acceso de la aplicación al servicio *Google Docs*, un enlace para revocar dicho permiso. El aspecto que presenta dicha página se muestra en la figura C.3.

Figura C.3: Página principal para usuarios administradores.

C.2.2. Crear y añadir un formulario

Para añadir un formulario a la aplicación, lo que debemos hacer en primer lugar es crearlo en *Google Docs*.

Crear un formulario en *Google*

Si todavía no hemos creado el formulario en *Google*, podemos hacerlo pulsando sobre el botón de crear un nuevo formulario en *Google*.

Figura C.4: Botón de crear un nuevo formulario en *Google*.

Dicho enlace nos llevará a una página de *Google Docs* en la que deberemos estar identificados con nuestra cuenta de *Google*, por lo que si no estábamos identificados se nos pedirá el acceso. Cuando estemos identificados aparecerá la página de creación de formularios cómo se muestra en la figura C.5.

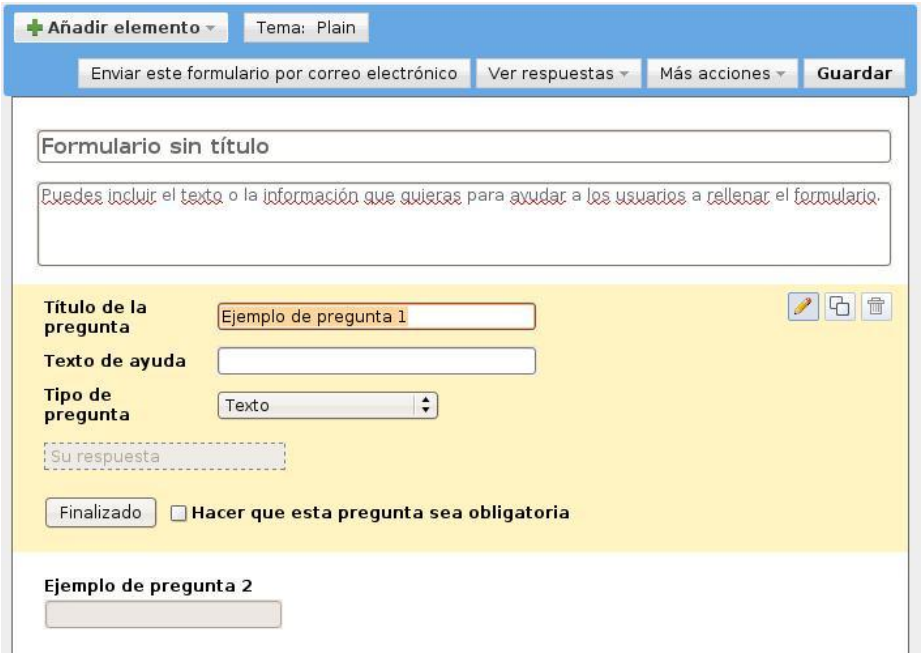


Figura C.5: Página de creación de formulario en *Google*.

En dicha página debemos añadir todas las preguntas que queremos que tenga nuestro formulario, y además añadir una pregunta adicional para llevar a cabo la identificación del usuario. Por comodidad se recomienda que dicha pregunta sea de texto, aunque se ha comprobado el correcto funcionamiento con otro tipo de preguntas.

Una vez creadas todas las preguntas del formulario, debemos pulsar sobre el botón de guardar, cómo se muestra en la figura C.6.

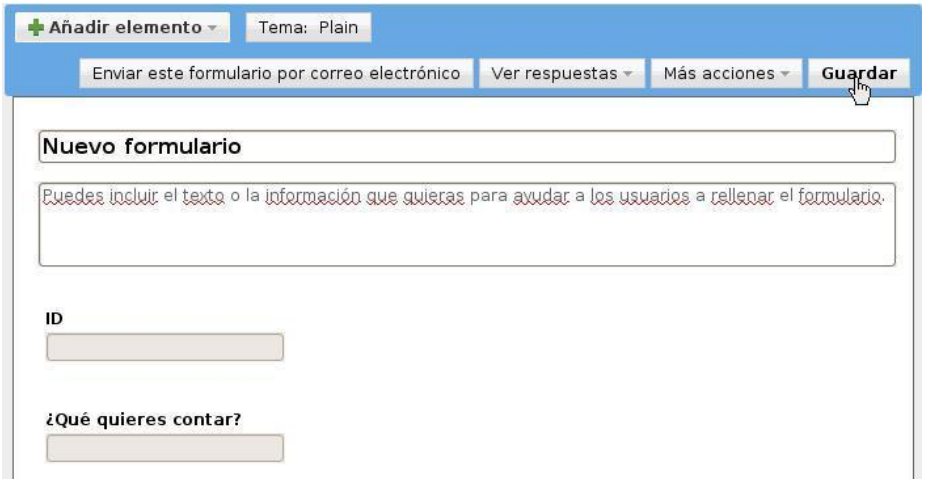


Figura C.6: Guardar el formulario en *Google*.

Cuando ya hemos guardado el nuevo formulario en *Google*, debemos de volver a la página principal de la aplicación *Gforms*.

Añadir un formulario a la aplicación

En función de si conocemos o no del formulario su *URL* y su pregunta de identificación de usuarios, utilizaremos dos distintos enlaces para acceder a la página de añadir un nuevo formulario a la aplicación.

- Si hemos copiado la *URL* del formulario y el nombre de la pregunta cuyo campo se va a utilizar para la identificación, podemos pulsar sobre la opción de añadir formulario a partir de *URL*:

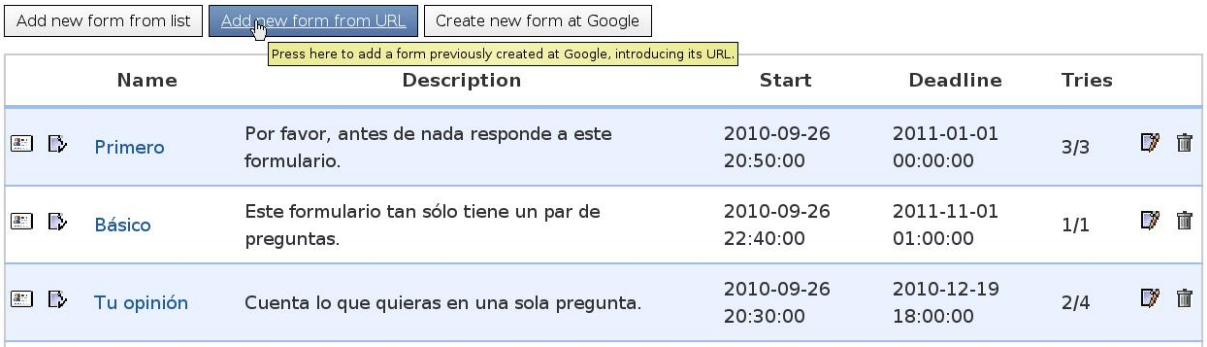


Figura C.7: Añadir formulario a partir de *URL*.

- Si por el contrario preferimos seleccionar el formulario de la lista de formularios de nuestra cuenta, deberemos pulsar en el enlace de añadir formulario a partir de la lista:

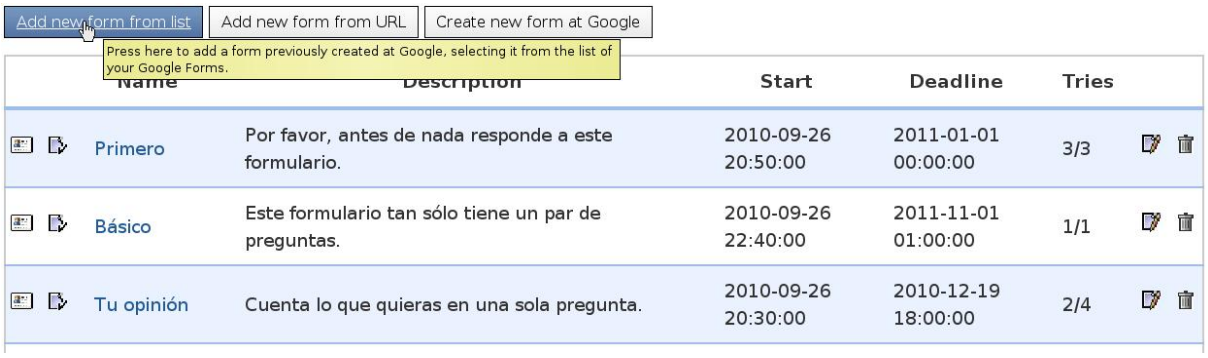


Figura C.8: Añadir formulario a partir de lista.

Si nuestra aplicación no está autorizada para acceder a nuestra cuenta de *Google* se mostrará una página solicitando dicho permiso, cómo se explicará en la sección C.2.7.

Cuando la aplicación tenga permiso para acceder a nuestra cuenta de *Google Docs* se mostrará la lista de formularios existentes en dicha cuenta, junto con la lista de preguntas de cada formulario:

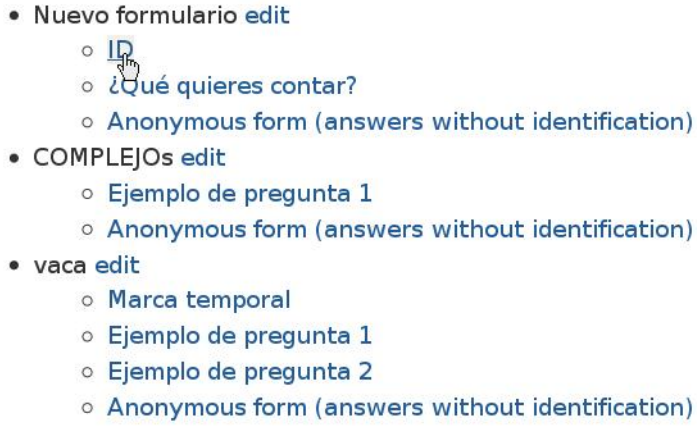


Figura C.9: Lista de formularios.

Deberemos elegir el formulario y la pregunta que queremos utilizar cómo identificador de usuario, y pulsar sobre su enlace correspondiente. Si queremos que el formulario no use autorización, deberemos pulsar sobre la opción de formulario anónimo.

A continuación accederemos a la página de añadir un nuevo formulario a la aplicación. En ella deberemos introducir los parámetros que deseamos que tenga el cuestionario, cómo se muestra en la figura C.10.

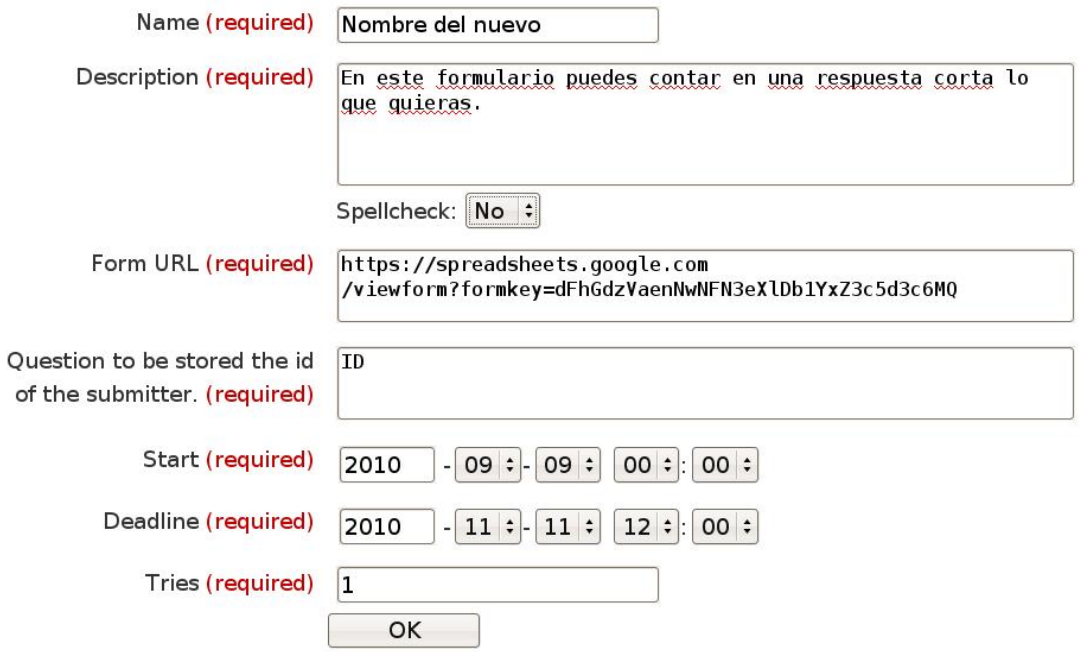


Figura C.10: Añadir formulario a la aplicación.

Si queremos que el formulario no tenga límite de intentos podemos introducir en dicho campo el valor '999', y si queremos que no tenga fecha límite, introducir cómo fecha el año '9999'. Para utilizar el formulario cómo anónimo y que las respuestas no tengan identificación de su autor deberemos dejar vacío el texto pregunta.

Tras pulsar el botón de aceptar, el formulario aparecerá en la lista, estando disponible para su realización para todos los usuarios de la comunidad mientras se cumplan los requisitos de plazo y número de intentos.

C.2.3. Ver las estadísticas de las respuestas de un formulario

Para ver las estadísticas de las respuestas de un formulario, primero debemos tener activada la opción correspondiente en *Google Formularios*. Para ello, en el menú de modificación de formularios en *Google*, debemos activar la casilla 'Publicar resumen de respuestas', la cual se encuentra pulsando sobre la opción 'Modificar confirmación' del menú 'Más acciones'.

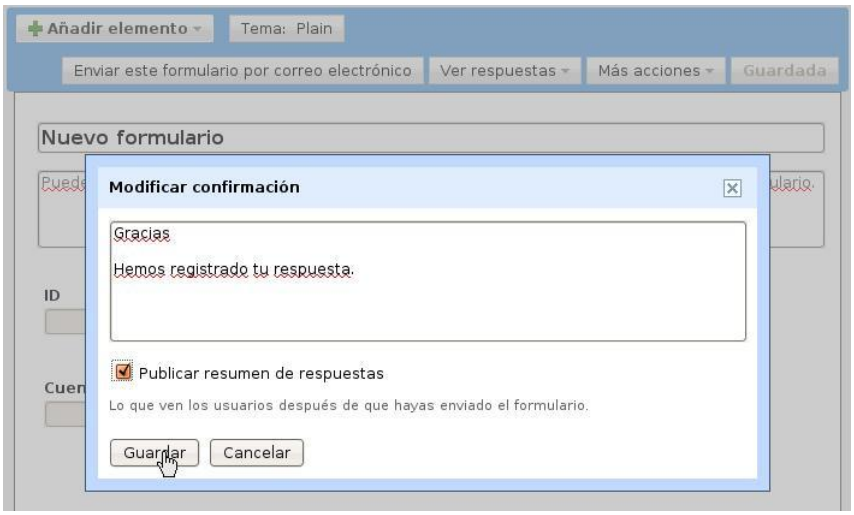


Figura C.11: Publicar estadísticas de respuestas.

Estando activada dicha opción, podremos ver las estadísticas de respuestas de un formulario a través de la lista de formularios, pulsando sobre el icono que aparece a la izquierda del nombre del formulario. Este icono se muestra en la figura C.12.

Figura C.12: Icono de estadísticas.

Para que dichas estadísticas no incluyan la de las respuestas no autenticadas, es recomendable en primer lugar realizar la acción de filtrar (ver C.2.4).

Se mostrará una página cómo la que aparece en la figura C.13.

A diferencia del servicio de *Google Formularios*, en la aplicación nunca se mostrará el enlace a las estadísticas de respuestas, por lo que los usuarios no podrán acceder a dicha página de acceso público mientras no descubran por otros medios su correspondiente *URL*.

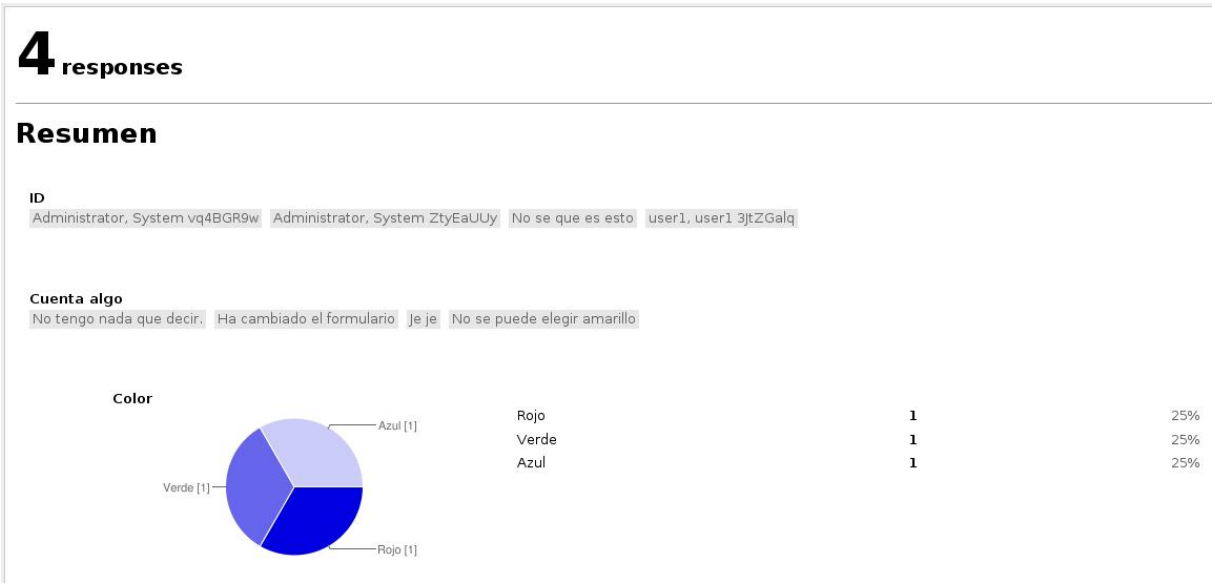


Figura C.13: Editar parámetros de un formulario.

C.2.4. Ver la lista de respuestas autenticadas

Para ver las respuestas que los usuarios de la comunidad han realizado al formulario, debemos pulsar sobre el icono de la figura C.14.

Figura C.14: Ver estadísticas de respuestas de un formulario.

Si nuestra aplicación no está autorizada para acceder a nuestra cuenta de *Google* se mostrará una página solicitando dicho permiso, cómo se explicará en la sección C.2.7.

Cuando la aplicación tenga permiso para acceder a nuestra cuenta de *Google Docs*, si se ha utilizado una pregunta del formulario para identificar a los usuarios (es decir, si no es anónimo), se mostrará la lista de respuestas, de manera similar a como aparece en la *Spreadsheet*, pero sin la columna correspondiente al campo de identificación, y con una columna inicial en la que aparecerá el nombre del usuario que realizó la respuesta el cual será un enlace a su dirección de correo electrónico.

La figura C.15 muestra la lista de respuestas autenticadas del formulario cuya *Spreadsheet* se muestra en la figura C.16.

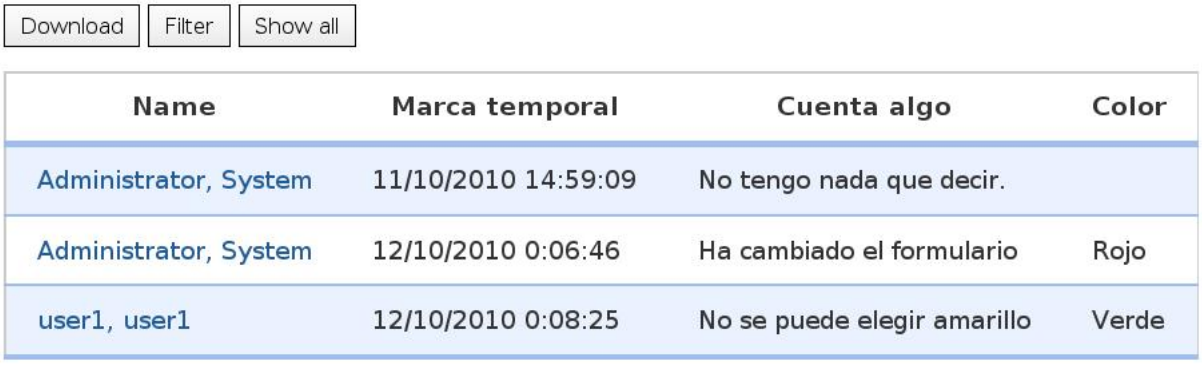


Figura C.15: Lista de respuestas autenticadas.

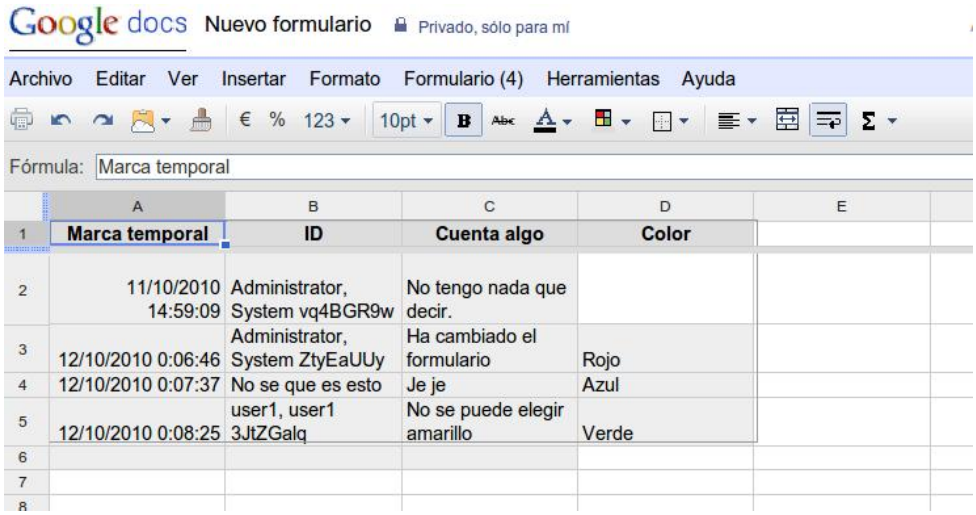


Figura C.16: *Spreadsheet* de ejemplo.

Además, desde la página que muestra la lista de respuestas autenticadas podemos realizar tres funciones distintas, pulsando en sus enlaces correspondientes que aparecen encima de la lista. Dichas funciones se describen a continuación.

Descargar la lista de respuestas autenticadas

Descarga un fichero en formato *CSV* con la lista de respuestas autenticadas.

Filtrar de la *Spreadsheet* los registros no autenticados

Elimina de la *Spreadsheet* los registros de respuestas no autenticadas, es decir, no realizadas a través de la aplicación por los usuarios de la comunidad. Esta opción es irreversible, por lo que

no se pueden recuperar los registros eliminados.

Mostrar las respuestas no autenticadas en la misma lista

Esta opción muestra en la misma lista los registros de respuestas no autenticadas, indicándolas cómo tal en la primera columna. Para la *Spreadsheet* de la figura C.16, la lista sería la mostrada en la figura C.17

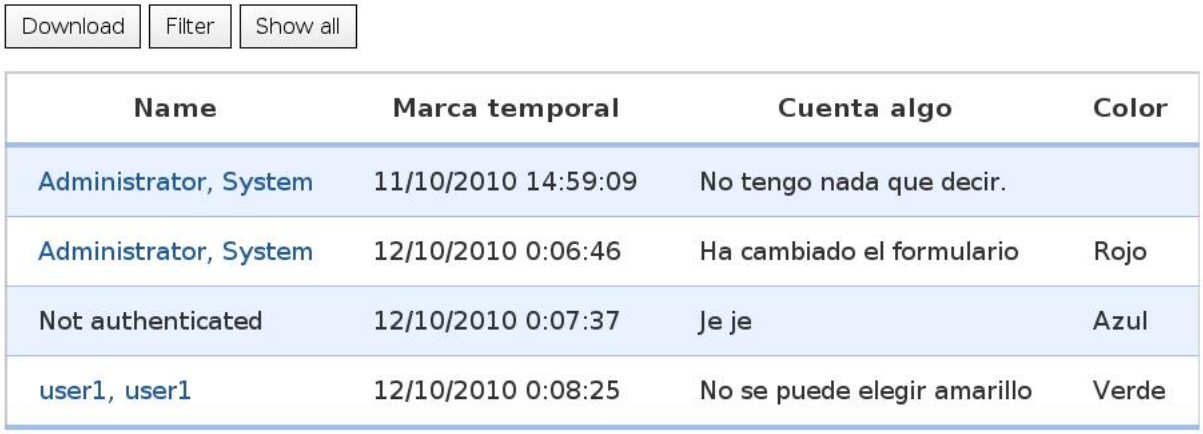


Figura C.17: Lista de todas las respuestas.

C.2.5. Editar los parámetros de un formulario

Es posible cambiar todos los parámetros de un formulario a excepción de su *URL*, ya que los formularios en la aplicación están asociados a un formulario en *Google*. Podemos cambiar su nombre o descripción sin restricción alguna, no hay problema en que dos formularios tengan el mismo nombre.

Si cambiamos el parámetros de plazo de realización debemos tener en cuenta que la fecha límite posterior para su realización debe de ser posterior a la actual, por lo que para cerrar el plazo de realizar formularios debemos introducir una fecha que sea unos minutos posterior a la actual, no pudiéndose cerrar de manera inmediata. Si queremos eliminar la posibilidad de que se realicen más respuestas, con efecto inmediato, podemos acceder a dicha opción desde su *Spreadsheet* asociada en la página de *Google Docs*.

En el caso de cambiar la pregunta de identificación de los usuarios, debemos de tener en cuenta que sólo debemos realizar dicha opción si hemos cambiado el nombre de la pregunta en el formulario de *Google*, y la pregunta se sigue correspondiendo con la misma columna en la

Spreadsheet. Si no, todas las respuestas que se hayan recibido hasta el momento dejarán de estar correctamente identificadas, a menos que copiemos los campos de identificación en la *Spreadsheet* a la nueva columna.

Para editar los parámetros de un formulario debemos de pulsar sobre el icono que se muestra en la figura C.18, el cual se encuentra a la derecha del formulario.

Figura C.18: Icono de edición de formulario.

Tras pulsar en dicho icono, aparecerá una página similar a la de añadir un nuevo formulario, pero con el campo de *URL* deshabilitado (ver figura C.19).

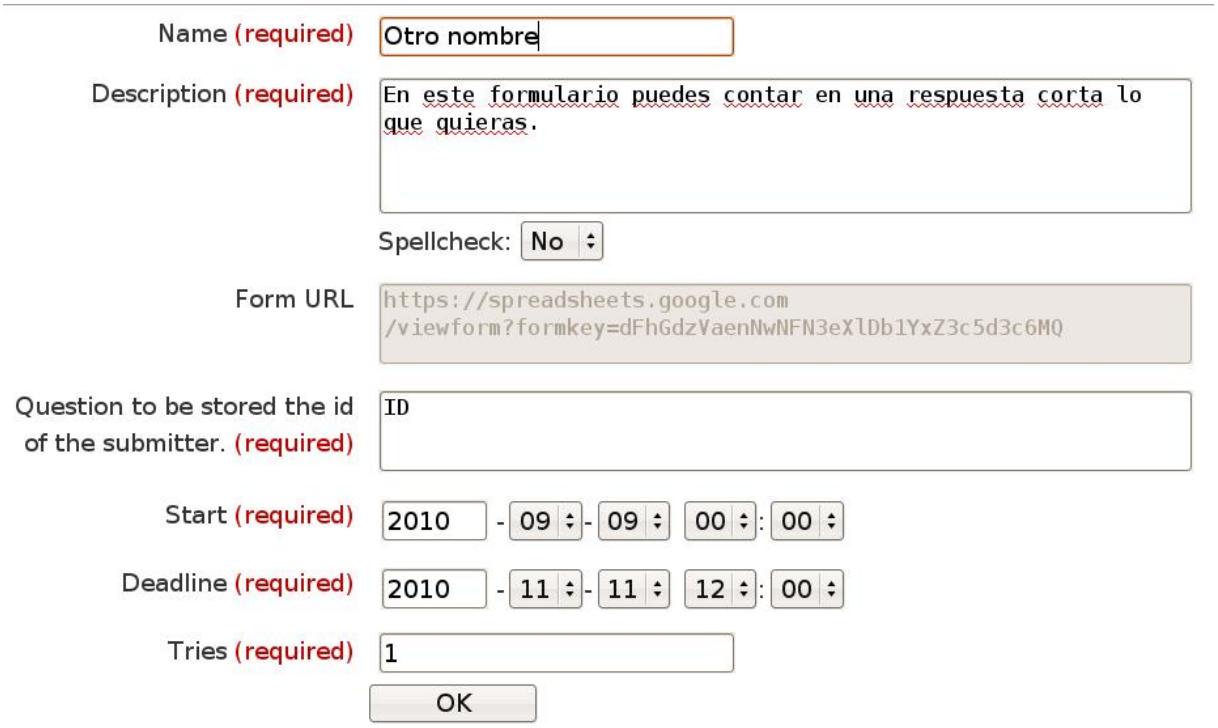


Figura C.19: Editar parámetros de un formulario.

C.2.6. Eliminar un formulario de la aplicación

Si eliminamos un formulario de la aplicación, se eliminarán todos sus parámetros asociados, junto con los registros de identificación de cada una de las respuestas realizadas a través de la aplicación, de manera irreversible.

Para eliminar un formulario, debemos de pulsar sobre su icono correspondiente, el cual aparece a la derecha del formulario. Dicho icono se muestra en la figura C.20.

Figura C.20: Icono de eliminación de formulario.

C.2.7. Autorizar a la aplicación a utilizar nuestra cuenta de *Google*

Si pulsamos sobre las opciones de añadir un formulario a la aplicación a partir de la lista de formularios, o mostrar las respuestas de un formulario, será necesario darle permiso a la aplicación para acceder a los datos de nuestra cuenta de *Google Docs*. En ambos casos, si en nuestro navegador no se ha iniciado una sesión en *Google* se nos pedirán los datos de acceso a nuestra cuenta. Cuando hayamos iniciado sesión en *Google*, se mostrará la página de la figura C.21.

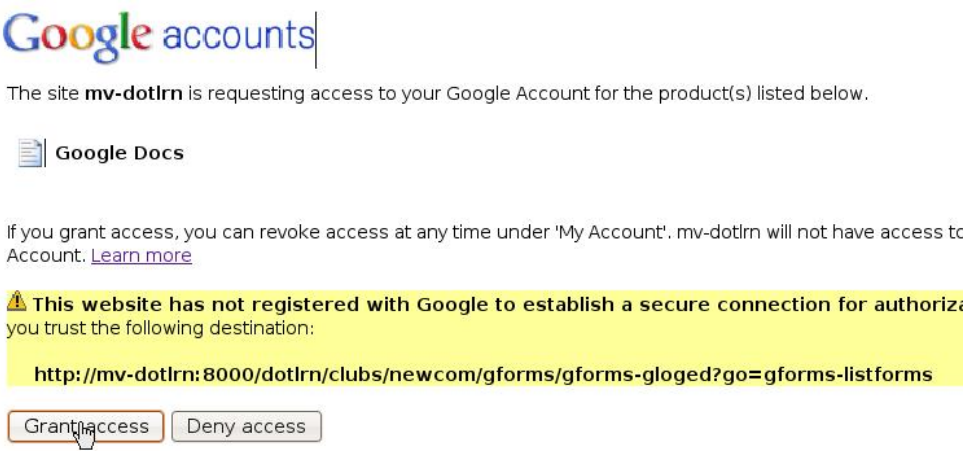


Figura C.21: Permitir el acceso a la cuenta de *Google*.

En dicha página deberemos de pulsar sobre el botón de permitir el acceso. A continuación se nos redireccionará a la página que queríamos acceder.

La página de petición de acceso a la cuenta de *Google* no se volverá a mostrar mientras el usuario mantenga su sesión en *.LRN*, a menos que pulse sobre la opción de revocar el *token*, la cual aparecerá una vez hayamos permitido el acceso en la página principal de la aplicación, debajo de la lista de formularios (ver figura C.22).

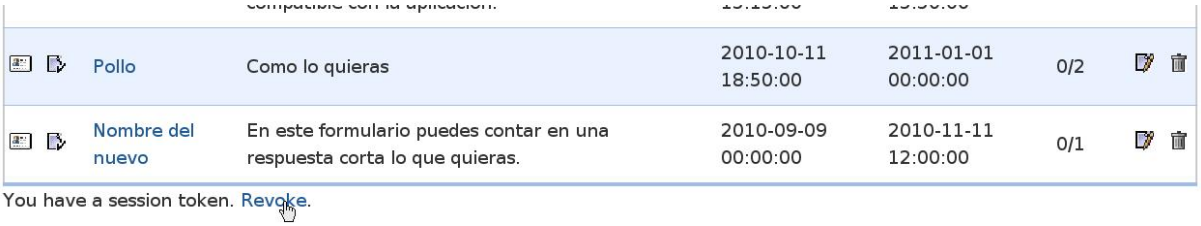


Figura C.22: Revocar el permiso de acceso a la cuenta de *Google*.

APÉNDICE D

Verificación de la aplicación

Para comprobar que no existen errores en la versión final de la aplicación, se han llevado a cabo una serie de pruebas que se detallan en este apéndice. Además, se ha solicitado a otras personas que prueben el funcionamiento de la aplicación, de manera que puedan descubrirse posibles errores que no haya considerado el programador. Durante el desarrollo del *software*, se ha comprobado el correcto funcionamiento de cada uno de los módulos que lo componen, sin embargo, dichas pruebas no se detallan en este apéndice debido a su gran número y escasa relevancia.

Cómo se comentó en la sección 3.3.6, se ha utilizado la herramienta Selenium para almacenar cada una de las pruebas, de manera que se puedan volver a realizar todas cada vez que se realice una modificación en el código de la aplicación.

Estas pruebas se realizaron sobre un ordenador con las siguientes características:

- Procesador *AMD Phenom* II X2 545.
- 2GB de memoria *RAM DDR3*.
- *Ubuntu* 10.04

Sobre este sistema, se ejecuta la máquina virtual *VMware* proporcionada por la Universidad Carlos III de Madrid[1], la cual tiene las siguientes características:

- *Xubuntu* 8.04
- *OpenACS* 5.4.3

[1] https://gradient.it.uc3m.es/xowiki/dotlrn_vm

- *.LRN* 2.4

Se han realizado las pruebas que se han considerado pertinentes para todos los usuarios posibles: usuario sin ninguna identificación, usuario identificado en el portal pero no perteneciente a la comunidad en la que se encuentra la instancia de la aplicación, usuario con identificación que pertenece a la comunidad pero sin permisos de administración, y usuario administrador de la comunidad.

La batería de pruebas para cada uno de estos tipos de usuario consiste en:

Usuario sin iniciar sesión en el portal

- Probar todas las páginas de la aplicación. Debe aparecer la página de identificarse en *.LRN*.

Usuario que no pertenece a la comunidad

- Probar todas las páginas de la aplicación. Un mensaje debe indicar que no se tiene permiso para verlas.

Usuario que pertenece a la comunidad pero no es administrador

- Comprobar que la página principal muestre todos los formularios disponibles en la instancia de la comunidad, y únicamente el enlace a la página que permite realizarlos. Es decir, no debe mostrar enlaces de las páginas de administración.

- Probar a realizar distintos tipos de cuestionarios con éxito. Esto incluye cuestionarios con distinto número de preguntas y de distintos tipos, siempre de una única página ya que son los que soporta la aplicación. Debe de registrarse el *hash* en la base de datos y aparecer en la *Spreadsheet*.

- Probar a realizar un formulario que haya sido eliminado de *Google Docs* pero no de la aplicación. Debe mostrarse un mensaje que avise de dicha circunstancia, en vez de mostrarse un mensaje de error en el código.

- Probar a realizar un cuestionario perteneciente a otra comunidad, tanto accediendo a `gforms-viewform` cómo a `gforms-response`. No debe permitirse su realización, mostrándose un mensaje de error.

- Probar a realizar un cuestionario antes del plazo (tanto con `gforms-viewform` cómo con `gforms-response`). Debe mostrarse mensaje de estar fuera de plazo.

- Probar a realizar un cuestionario pasado su plazo (tanto con `gforms-viewform` cómo con `gforms-response`), incluyendo el caso de iniciar el cuestionario dentro del plazo, pero enviarlo cuando ya ha terminado. Debe mostrarse el correspondiente mensaje de haber terminado el plazo.

- Probar a realizar un cuestionario un número de veces superior a las permitidas (tanto con `gforms-viewform` cómo con `gforms-response`). Debe mostrarse un mensaje que indique que se ha superado el número de intentos.

- Probar a acceder al resto de páginas (de uso exclusivo por los administradores) introduciendo directamente su *URL*. Deben mostrar mensaje de error o redireccionar a la página principal en el caso de las páginas que realicen dicha función para el caso de administradores (`gforms-revoke` y `gforms-delete`). Se comprueba que no se hayan producido cambios en la base de datos.

Usuario administrador

- Comprobar que la página principal muestra todos los formularios disponibles, y permite acceder a todas las opciones de cada uno. Además deben aparecer en dicha página las opciones de añadir un nuevo formulario (introduciendo *URL*, y a partir de la lista), crear nuevo formulario en *Google* y revocar el *token* en caso de que exista. Todos los enlaces deben funcionar correctamente.

- Comprobar el acceso a las páginas que requieren *token* (tanto `gforms-listforms` cómo `gforms-listsubmits`) cuando no se dispone de *token*. El proceso de obtención de token debe de realizarse correctamente, añadiéndose a la base de datos.

- Comprobar el acceso a las páginas que requieren *token* cuando éste es de otra sesión. Debe eliminarse de la base de datos y solicitar otro nuevo.

- Comprobar el acceso a las páginas que requieren *token* cuando éste no es válido. Debe eliminarse de la base de datos y solicitar otro nuevo.

- Comprobar que `gforms-listforms` funciona correctamente tanto cuando se dispone de *token* válido cómo cuando se solicita, mostrando todos los formularios disponibles con sus correspondientes preguntas. Los enlaces deben redireccionar correctamente a la página de creación de formulario en la comunidad, creándose el formulario correctamente.

- Comprobar que se puede añadir un formulario tanto utilizando el acceso a través de la lista de formularios, cómo introduciendo directamente la *URL*. Además, se debe comprobar que si se introducen datos no válidos, se avise de ello para su corrección, y posteriormente la creación se produzca correctamente. Los datos no válidos pueden ser: *URL* de formulario no existente, *URL* de formulario ya existente en la aplicación, fecha límite anterior a la actual o la de inicio y número de veces mayor que cero.

- Comprobar que `gforms-listsubmits` funciona correctamente tanto cuando se dispone de *token* válido cómo cuando se solicita, mostrando únicamente las respuestas del formulario realizadas mediante la aplicación. Para comprobar esto, se responderá varias veces al cuestionario utilizando su *URL* pública en lugar de acceder a la aplicación. Probar en primer lugar la opción de mostrar todas las respuestas para ver que aparecen las no autenticadas, indicándose cuáles son. A continuación se prueba a descargar el fichero *CSV* y se comprueba que contiene sólo las autenticadas. Por último se prueba la opción de filtrar, comprobando que elimina de la *Spreadsheet* todas las respuestas no realizadas mediante la aplicación. Al utilizar de nuevo la opción de mostrar todos, sólo aparecerán las autenticadas, al haberse filtrado las demás. Estas opciones también se probarán en distinto orden, para asegurar el correcto funcionamiento de esta página en todas las circunstancias.

- Comprobar que `gforms-listsubmits` funciona correctamente cuando no existen registros en la *Spreadsheet*.

- Comprobar que `gforms-listsubmits` funciona correctamente para formularios no autenticados, es decir, aquellos en que no existe ninguna pregunta con el texto indicado para dicho formulario.

- Comprobar que `gforms-listsubmits` funciona correctamente para formularios que no disponen de tabla (por haberse creado en una versión antigua de *Google Formularios*).

- Probar a borrar un formulario, y comprobar que desaparece de la base de datos, junto con

todas las respuestas de dicho cuestionario.

- Probar a editar un formulario. Deben de aparecer la *URL* cómo campo no modificable en el menú de edición (`gforms-edit`). Tras la edición, los parámetros modificados deben aparecer cómo tal en el menú de edición. Comprobar que se puede realizar un formulario al que se le había acabado el plazo, tras prolongar su plazo por encima de la fecha actual. Lo mismo con el número de intentos.

- Comprobar que funciona correctamente el enlace a la página de estadísticas de un formulario, aunque solamente se muestren en caso de estar activada dicha opción.

- Comprobar que la opción de revocar *token* aparece únicamente cuando existe en la base de datos y es de la sesión actual. Probar a realizar dicha opción y comprobar que es eliminado de la base de datos. Si se desea comprobar que la petición enviada a *Google* es respondida satisfactoriamente, debe consultarse en el fichero *log*, el código de respuesta de dicha petición, el cual debe de ser 200.

Bibliografía

[1] AOLserver ADP Development. http://www.aolserver.com/docs/devel/tcl/adp-overview.html Consulta: 18 oct. 2010.

[2] Git-Svn Comparison. https://git.wiki.kernel.org/index.php/GitSvnComparsion Consulta: 18 oct. 2010.

[3] Git vs SVN – Which is Better? http://www.looble.com/git-vs-svn-which-is-better/ Consulta: 18 oct. 2010.

[4] Tcl Core Development. http://www.tcl.tk/community/coreteam/ Consulta: 18 oct. 2010.

[5] Processing XML documents with Tcl and tDOM. *Linux Magazine*, 2002. http://www.linux-magazine.com/issue/20/tDOM.pdf Consulta: 18 oct. 2010.

[6] Desire2Learn to replace WebCT, Blackboard. Feb. 2004. http://media.www.spectatornews.com/media/storage/paper218/news/2004/02/02/CampusNews/Desire2learn.To.Replace.Webct.Blackboard-594170.shtml Consulta: 18 oct. 2010.

[7] Blackboard: Bully or Misunderstood? *Inside Higher Ed*, Aug. 2006. http://www.insidehighered.com/news/2006/08/18/patent Consulta: 18 oct. 2010.

[8] Patent Office Orders Re-Examination of Blackboard Patent. *Software Freedom Law Center*, June 2007. http://www.softwarefreedom.org/news/2007/jan/25/blackboard-reexam-ordered/ Consulta: 18 oct. 2010.

[9] Utah Network Crash Erases Student Data. *The Chronicle*, Nov. 2007. http://chronicle.com/blogPost/Utah-Network-Crash-Erases/3517 Consulta: 18 oct. 2010.

[10] Background Information About LMS Deployment from the 2008 Campus Computing Survey. 2008. http://net.educause.edu/ir/library/pdf/LIVE0914ccp.pdf Consulta: 18 oct. 2010.

[11] Bad News for Blackboard, Good News for Moodle. Mar. 2008. http://mfeldstein.com/bad-news-for-blackboard-good-news-for-moodle/.

[12] 7 Things You Should Know About LMS Alternatives (ID: ELI7062). July 2010. http://www.educause.edu/Resources/7ThingsYouShouldKnowAboutLMSAl/207429 Consulta: 18 oct. 2010.

[13] M. Bellare and T. Kohnoy. Hash Function Balance and its Impact on Birthday Attacks. May 2004. http://eprint.iacr.org/2003/065.ps Consulta: 18 oct. 2010.

[14] P. Cuesta. Web 2.0. http://www.slideshare.net/pedrocuesta/la-web-20-69471 Consulta: 18 oct. 2010.

[15] R. K. Ellis. A Field Guide to Learning Management Systems. 2009. http://www.astd.org/NR/rdonlyres/12ECDB99-3B91-403E-9B15-7E597444645D/23395/LMS_fieldguide_20091.pdf Consulta: 18 oct. 2010.

[16] M. Feldstein. Blackboard Patents the LMS. July 2006. http://mfeldstein.com/blackboard_patents_the_lms/ Consulta: 18 oct. 2010.

[17] M. Feldstein. Blackboard Is Losing Customers, but What Does It Mean? Nov. 2007. http://mfeldstein.com/blackboard-is-losing-customers-but-what-does-it-mean/ Consulta: 18 oct. 2010.

[18] M. Feldstein. Blackboard's Market Share Erosion. Mar. 2009. http://mfeldstein.com/blackboards-market-share-erosion/ Consulta: 18 oct. 2010.

[19] D. R. García. Proyecto de Evaluación de plataformas de Teleeducación para el ámbito universitario. Sept. 2003. http://www.uv.es/ticape/docs/dario/mem-dario-v8.pdf Consulta: 18 oct. 2010.

[20] P. Greenspun. Introduction to AOLserver, Part 1. 1999. http://philip.greenspun.com/wtr/aolserver/introduction-1.html Consulta: 18 oct. 2010.

[21] J. Jordan. Portal Problems, Blackboard Blackout. *The Hunter Envoy*, Feb. 2009. http://media.www.thehunterenvoy.com/media/storage/paper1327/news/2009/02/18/News/Portal.Problems.Blackboard.Blackout-3635795.shtml Consulta: 18 oct. 2010.

[22] S. E. Lakhan and K. Jhunjhunwala. Open Source Software in Education. June 2010. http://www.educause.edu/EDUCAUSE+Quarterly/EDUCAUSEQuarterlyMagazineVolum/OpenSourceSoftwareinEducation/162873 Consulta: 18 oct. 2010.

[23] P. Lengyel, M. Herdon, and R. Szilágyi. Comparison of Moodle and ATutor LMSs. 2006. http://www.mtakpa.hu/kpa/download/1203937.pdf Consulta: 18 oct. 2010.

[24] M. Molist. Moodle. Institutos y universidades apuestan por esta plataforma libre de 'e-learning'. http://www.aulaintercultural.org/article.php3?id_article=1517 Consulta: 18 oct. 2010.

[25] Z. Rosen. Sakai vs. Moodle. http://www.zacker.org/sakai-project-vs-moodle Consulta: 18 oct. 2010.

[26] O. C. Santos, J. G. Boticario, E. Raffene, and R. Pastor. Why using dotLRN? UNED use cases. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.135.7263&rep=rep1&type=pdf Consulta: 18 oct. 2010.

[27] E. W. Weisstein. Birthday Problem. *MathWorld–A Wolfram Web Resource*. http://mathworld.wolfram.com/BirthdayProblem.html Consulta: 18 oct. 2010.